Peer-reviewed version available at *Cryptography* **2018**, *2*, 10; doi:10.3390/cryptography2020010

Article

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

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

[1] Auckland University of Technology; alan.litchfield@aut.ac.nz
[2] Cybercraft; jeff.herbert@cybercraft.net
* Correspondence: alan.litchfield@aut.ac.nz
† These authors contributed equally to this work.

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 1970's 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. Typically, methods to maintain control of copyrighted software are software activation using a paper based key code (such as on software packaging), software license validation through online registration [1] and hardware devices [2] such as dongles. In general, smaller vendors implement software validation in the form of an activation key, large publishing houses such as Microsoft and Adobe use proprietary centralised software license validation services via the Internet as the primary medium.

Due to a combination of technological and economic changes, software license validation is growing in complexity. Commercial models for software sales and distribution are more complex with multiple parties in the supply chain including software owners, multiple levels of distributors and customers [3]. Similarly, software is increasingly complex as the scope of use increases [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 novel solution to the problem of illegal redistribution of software licenses through a method of decentralised, peer-to-peer, publicly auditable software license validation. The method utilises cryptocurrency blockchains similar to Bitcoin that could be used by anyone from an independent software writer to a large software vendor.

The paper is structured as follows: (i) A summary of software piracy issues demonstrating the continued growth of software piracy is provided, especially in the mobile computing environment; (ii) 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; (iii) cryptocurrency blockchain functions are described and the benefits of a decentralised peer-to-peer architecture are presented, and (iv) High level and reference architecture are described for the ReSOLV system for licensing software.

2. Related Work

In this section are presented 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.

2.1. Software piracy

Software piracy commonly describes the copying or use of computer software in violation of its license and has been a long standing technology issue since the advent of 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. Further, 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 [15,16,17]. 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 follows: (i) End User Piracy where a company employee reproduces copies of software without authorisation; (ii) Client-Server Overuse, when too many users on a network are using a central copy of a program at the same time; (iii) Internet Piracy, when software is illegally downloaded from the Internet; (iv) Hard-Disk Loading occurs when a business that sells new computers loads illegal copies of software onto hard disks; (v) Software Counterfeiting is the illegal duplication and sale of copyrighted material with the intent of directly imitating the copyrighted product.

The Internet provides a convenient medium for software piracy by enabling the illicit download of copyright software and the borderless nature of the Internet means that it is difficult to enforce ownership in all jurisdictions. The BSA estimates that in 2013, 43% of software on home computers around the world was not properly licensed, with a commercial value of US$62.7 billion. Even subscription based models such as that provided through cloud computing are not expected to provide a significant impact on reducing software piracy with 52% of online credentials being shared [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] state that 84 of the top 100 iOS applications are affected by piracy. Android developers experience a particularly high and consistent 90%–99% piracy rate over the last two years. Given these figures, the impact on the developer community must be significant [21,22,23]. Additionally, pirated mobile apps present a new attack vector for criminals that upload pre-infected files for unsuspecting users to download [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; the piracy rate on mobile devices appears higher than on desktop computers. 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

A variety of methods to protect the copyright of software authors have been in place since the early 1980's. For example, Suhler *et al.* [8] suggest that to be successful, software authorisation (validation) needs to be inexpensive, compatible with other systems and easy to implement. Similarly, Morgan and Ruskell [2] present various practical measures to deter or prevent unauthorised copying. However, feasibility is dependent upon the cost of the measure versus the perceived value of the software. The three primary methods for software license authorisation are copy protection, software validation using a paper-based key and hardware-based keys. The limited scope for authorisation methods on early desktop computing platforms was due to the relatively high cost of hardware devices, limited processing power on PC's, restricted opportunities for encryption methods, and a form of software license validation media that was easy to distribute and deploy. 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 code modification to remove software license validation methods, 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]. However, 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 exist [24, for example]. However in the Apple iOS ecosystem Apple developers must undertake an App Review process to ensure their iOS App complies 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, for example]. 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 make the claim for a client-server software license validation method using an Internet-based centralised database for software license validation. Their system also detects and records the hardware platform characteristics that the software is installed on. Whereas Khan *et al.* [20] describe a framework for mobile application piracy control, Pirax, that 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 deters 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. Enterprise software license models for sales and distribution are becoming ever more complex with multiple parties in the supply chain including software owners, multiple levels of distributors and customers [3]. Consequently, complying with software licensing is more difficult as the scope and complexity of software use increases. That is, feature specific enablement keys for software packages, geographical distribution of software deployment, size of customer organisation and user base, the shift to SaaS models and an increasing use of 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, the BSA has established Verafirm [15] that provides SAM tools and solutions for Small and Medium Enterprises (SME's) and enterprises to assist organisations with the management of software licensing. 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.

As mentioned, Khan *et al.* [20] describes a framework, Pirax, for mobile App piracy using a cloud-based license server. When an App is first run on a mobile device, the App needs to register with the cloud license server to receive a unique license. This is achieved by creating a "node locked licensing scheme." The scheme utilises the device's unique International Mobile Equipment Identity (IMEA) number and the cloud server Universally Unique Identifier (UUI) to generate a unique registration serial number and an activation serial number for the App license. Each time the App is run, it verifies its encrypted activation serial number with the cloud license server, and if validation is successful, the App is allowed to run. It allows Apps to be authorised, or revoked, at the cloud license server. Khan *et al.* [20] states that the Pirax framework mitigates several types of hacking, including preventing the App executing on a cloned image of the phone, and defeating a hacker extracting the App and executing it on another compatible platform.

Veerubhotla and Saxena [32] describe a theoretical common framework to protect digital objects, software libraries, and controlled software execution, with support for both online and offline models. The framework outlines 3 actors: (i) The publisher that produces a digital artefact to be protected; (ii) the consumer with an encrypted artefact downloaded from the publisher; and (iii) a DRM server that encrypts the publisher's artefact (and returns it to the publisher) and validates user license requests. The publisher needs to create a library for the core functions of the Digital Rights Management (DRM) functionality, and distributes it with the software. This becomes a bootstrap for the encrypted software, as well as being available after the software is decrypted. The confidentiality of artefacts and transport utilise a combination of X.509 certificates, and asymmetric and symmetric cryptographic keys, to support online and off-line capability. Veerubhotla and Saxena [32] state several benefits for prevention of piracy, including secure communications, tamper-proof licenses, protection of cryptographic keys, and protection of software libraries and applications. Several non-functional benefits are also stated. However, these are primarily generalised assumptions reflecting architecture, and are not supported in the paper.

Tagged Transaction Protocol (TTP) is a software license provenance model created and defined by Palmer [29]. TTP enables the anonymous transition of licenses across a typical supply chain actors, for example supplier, reseller, and customer. The provenance model defines a set of terms and relationships to record the provenance of items in reseller chains, where actors in the supply chain are granted control at each stage. As each item moves through the supply chain, it is tagged. The Tag Generation Centre is responsible for creating tags and securely transmitting them to each of the actors in the supply chain. Palmer [29] states several benefits of TTP including prevention of spoofing, fabrication, cloning, identity revelation, linkability and protection from network sniffing.

A method proposed by Fortin [33], the Master Bitcoin Model (MBM), demonstrates provenance using cryptocurrencies. A proof-of-concept subsequently created by Lebo [34], "Dissent," uses Namecoin [35] to demonstrate license provenance. This model uses the immutable nature of the cryptocurrency blockchain and the cryptographic proofs to validate that a user owns a specific bitcoin that is linked to a license. It does not, however, provide any license validation in itself; it simply provides proof of ownership of 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. Requirements for a software license validation method

Suhler *et al.* [8] and Morgan and Ruskell [2] stress that a successful software licence validation method needs to be inexpensive, compatible with other systems, easy to implement, and relevant to the value of the software. In addition, to be effective against software piracy a successful software license validation method requires several premises to be met: (i) The license mechanism needs to be hard to copy; (ii) rights to software licenses need to be easily validated; (iii) software licenses cannot be repeatedly generated; (iv) validation needs to protect from Man-in-the-Middle attacks; and, (v) protection against reverse engineering and code modification to remove software license validation methods. Therefore a mechanism that can generate unique values that can't be regenerated but can be easily verified against a source engine at any time is required. In addition to these requirements, an effective mechanism needs to be able to handle complex enterprise licensing schemes.

To recap, software piracy and software license validation has been an issue since the late 1970's and 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 cryptocurrencies such as Bitcoin that provide essential building blocks for software license validation. Bitcoins are represented as cryptographically validated digital signatures and are therefore unfeasible to copy whilst decentralised transaction features ensure a bitcoin digital signature cannot be repeatedly generated and used . Finally, bitcoin transactions are cryptographically secure using public key cryptography to prevent Man-in-the-Middle type attacks.

Therefore, to create a method for software license validation mechanism we present a cryptocurrency blockchain ecosystem that meets the requirements for software license validation. The following sections introduce the cryptocurrency, its underlying blockchain technology and apply blockchain concepts to demonstrate how to achieve a fully distributed software license validation method that can be deployed globally.

3. Cryptocurrencies and the blockchain

Developed by Nakamoto [36] and first introduced with the creation of Bitcoin, cryptocurrencies are a new form of virtual currency. A cryptocurrency is a purely decentralised peer-to-peer electronic cash system and is the first technology to successfully overcome the requirement for a centralised party to validate transactions. The cryptocurrency architecture provides several

blended features that include cryptographic validation for all transactions, decentralised money, mint and transaction processing functions, all stored on public ledgers within a quasi-anonymous framework [37]. Cryptocurrencies use public-key cryptography to validate transactions between all participants and digital signatures to ensure transactional integrity and non-repudiation [38]. The cryptographic mechanisms used by cryptocurrencies provide for strong confidentiality, data integrity and non-repudiation services that are in use by business, government and military organisations globally . In a cryptocurrency ecosystem, the public key can be considered as the participant's account number whilst the private key represents the participant's ownership credentials. All participants have digital wallets that are used to store private keys as well as digital signatures that represent cryptocurrency entitlements (bitcoins) that the participants own. Wallets can be stored privately or online on websites or in exchanges depending on the requirements of the participant.

Needing to be resilient to threats and attacks as well as being a stable and liquid currency, cryptocurrencies have yet to prove their robustness both technologically [39,40] and economically [41,42]. However as developers seek to use cryptocurrencies in more practical applications, it is the cryptocurrency architecture, the blockchain, that is the point of development of new cryptocurrency based applications [43,44,45].

3.1. Economics

The objective of a cryptocurrency is to overcome the necessity of a centralised "trusted authority" [36] and thus remove or significantly reduce transaction fees associated with transactions such as those incurred with commercial banking transactions. As a peer-to-peer decentralised technology, cryptocurrencies historically rely on a network of low cost computers running software that perform primary functions within the cryptocurrency ecosystem[1]. The computers running this software are "Miners" that create bitcoins, validate bitcoin transactions and maintain the integrity of the blockchain public ledger. Miners are rewarded for their investment in running the bitcoin software by creating the bitcoins themselves. Miners may also (but not always) receive a transaction fee for validating bitcoin transactions. For example, participants in the Ripple [46] and Gridcoin [47] cryptocurrencies run transaction validation software on a voluntary non-profit basis in return for being allowed to provide compute and storage resources to run the mining software.

Most cryptocurrency ecosystems have a fixed number of bitcoins that can 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. To function as a financial instrument and with an unlimited supply of bitcoins, some cryptocurrencies such as Peercoin [48] are established on an inflationary economic model. It is assumed that these approaches lead to cost effective cryptocurrency and blockchain ecosystems through lower transaction fees [49]. Furthermore, McCook [50] makes the claim that cryptocurrencies are found to incur considerably lower cost than fiat currencies when compared to economic, environmental and socioeconomic factors.

3.2. Centricity

Centricity is the characteristic that makes the cryptocurrency ecosystem of an attractive proposition to stakeholders. Baran [51] identifies three types of network centricity: centralised, decentralised, and distributed. While the global fiat currency banking system is centralised, each type is used in cryptocurrency systems to validate currency transactions. Most cryptocurrencies

---

[1] The term ecosystem used in this context does typically mean the usual portmanteau of "ecological system" but instead evokes a new portmanteau, "economic system". That is, 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.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 March 2018 doi:10.20944/preprints201803.0217.v1

Peer-reviewed version available at Cryptography 2018, 2, 10; doi:10.3390/cryptography2020010

are distributed, offering no central point of failure and no master node required for the transaction validation process. A few cryptocurrencies, such as Ripple, are decentralised, with a master node for transaction validation.

3.3. Transactions

Bitcoin transactions are defined as a message between participants and consists of 3 segments: (i) Signature, the originator's digital signature signed with the originator's private key so that other Bitcoin nodes can verify that the message came from the originating participant; (ii) Inputs, a list of the signatures of transactions already in the ledger where the originator was the recipient of bitcoins and the input bitcoins are the funds the originator uses in a transaction; (iii) Outputs, a list of how the funds in the inputs are to be distributed. All the funds input must be redistributed as outputs, so the originator will pay the recipient the required amount and the remainder is returned as change.

A transaction must have exactly the same number of bitcoins in input and output lists (Figure 1). Hence if user U1 has 10 bitcoins and wants to send 2 bitcoins to user U2, then the transaction will result in U1 receiving 8 bitcoins and U2 receiving 2 bitcoins. Note that if the transaction occurs with an unequal result, then the transaction fails to complete with a valid or consistent state.

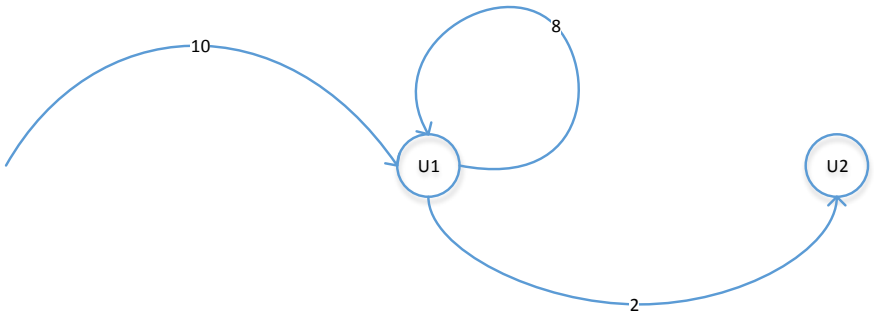

Figure 1. Transaction input and output.

The recipient is identified as a unique bitcoin address derived from the public key, so cryptocurrency transactions can be traced throughout the blockchain to the beginning of the creation of that specific cryptocurrency. Thus a mechanism for checking the ownership of cryptocurrency bitcoins is formed. Problems associated with double spending are avoided through publicly verifiable ledgers and thus a high degree of certainty for the participants of the cryptocurrency ecosystem is also provided.

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 [36], Proof of Stake [52,53], plus any other proofs that may be required. Blocks are created at regular intervals, for example every 10 minutes, and each block contains the following properties:

1. Transactions (T) sent between participants
2. Proof of Work (P_w), a unique digest created when a new block is discovered, and
3. Reference (R) to the digest of the previous block.

Thus the block (B) is represented as

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

Wood [54] defines a block as a collection of transactions that are chained together with a cryptographic hash that operates as a reference mechanism. A block functions "as a journal recording

a series of transactions together with the previous block and an identifier for the final state" [2015, p 2] although Wood states that the journal does not record the final state due to size constraints. He says too that 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" [2015, p 2]. The key to successful mining is reliant upon a cryptographically secure proof.

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 [55]. 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.* [56] define P_w as the means by which confidence is given to a bitcoin as a consequence of it being difficult to replicate, 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. Thus when a transaction completes, the event is broadcast on the cryptocurrency network for verification by multiple nodes (miners). This verification process is designed to prevent double spending of the bitcoin. When each node has received the transaction, the node tests the validity of the transaction. The more nodes that accept the transaction, the less likely it is that the transaction represents a double spend.

Therefore, a transaction may end with one of three states:

1. Confirmation of the successful completion of a transaction, which does not occur until 6 blocks have confirmed the transaction.
2. Unsuccessful completion of the transaction, for example if the immediate responding nodes reject the transaction. Otherwise the transaction has timed out after a period, typically 60 minutes for Bitcoin.
3. Unverified transactions are placed in an unverified transaction bucket, and will be inserted into the next block once it is created.

The current new block then has a new value P_w^1 inserted as its Proof of Work. When a block is verified the value, P_w^1, is inserted into the next block. It is this sequence of $\{ P_w^1, P_w^2, \ldots, P_w^n\}$ that defines the unique blockchain and its associated record in the ledger. In the next section is described how blockchain characteristics are applied to a decentralised software validation method.

4. Decentralised software license validation

Comparing the proposed methods above, most methods have functions such as: (i) The capability to generate a unique license for the client actor; (ii) utilise local device unique identifiers to prevent an App from executing on unauthorised devices; and (iii) use public/private key encryption to protect the license, to validate the user, and to prevent the license from being copied. All methods use the Internet for ubiquity.

However, each method has key differences that limit its effectiveness as a Software License Validation (SLV) method. Pirax targets the mobile computing device platforms, has "node-locked licensing", and does not require any encryption of the App. The DRM Framework claims to provide piracy prevention for cross-platform and multiple content formats, but requires publishers to encrypt the application or content, and stores decryption keys on the server. TPP focusses on supply chain

license provenance, providing license distribution and actor anonymity, but by nature is a one way method and does not protect from piracy once the client actor receives the license. The Bitcoin method is fully distributed and provides anonymity, but requires a bootstrap application to access the blockchain, and only validates an entitlement to a license, it does not store any client actor or license metadata.

The first common element across all methods is the requirement for the user (or device) to prove that they are the actual entity, and validate themselves through a unique identifier, such as a private key or an embedded unique identifier. The second common element, is a license key that is bound to the validated user to establish entitlement. Together these form the entitlement for use of the software.

In describing a decentralised SLV method, we use the term bitcoin as a generic descriptor for a coin from an existing cryptocurrency. The principle of decentralised SLV is to use bitcoins held by the owner to represent entitlement to software. Thus we do not attribute any economic value to a coin but see it as 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. The blockchain may store a record of all software licenses possessed by an end user. A software developer or vendor can allocate licenses to users easily and cost effectively via a decentralised peer-to-peer blockchain architecture.

5. ReSOLV Model

In this section the ReSOLV model is presented, with the main constructs for SLV on the blockchain and functional and non-functional requirements for the ReSOLV model through Requirements Engineering (RE) [57]. Due to the multiple actor interactions in the distributed ecosystem, functional and non-functional requirements are identified. The RE process comprises a set of activities to create consensus among stakeholders and establish a requirements document that satisfies stakeholder requirements [58]. Requirements elicitation focusses on constructing a model that derives its functions from earlier non-blockchain based SLV work, for example the MBM.

The MBM, as a basic form of SLV, uses Namecoin as the underlying blockchain. In this model, the vendor address/bitcoin combination represents license ownership, and if the user has a transaction that shows the bitcoin originated from a specific vendor address, then the user is considered to have ownership of the software. Although the MBM presents an opportunity for a chain-of-title based software licensing system, there are disadvantages that will deter use in the software industry. The MBM is essentially a binary system; the end user either has entitlement to run the software, or they don't. Although it is possible to create multiple Master bitcoins to represent individual software features for activation, this creates an overhead for each software feature, and there is no capability to include additional functions that are commonly used by software vendors, such as multi-user licensing for organisations, and management of corporate and private licenses. Furthermore, it is possible to quickly move Master bitcoins between end users, defeating the intent of the authorisation method that the MBM intended.

By comparison, the ReSOLV Model uses a tokenised approach to validation whilst still referring to a Master bitcoin. This approach offers improvements over the centralised software validation model, permitting the model to be applied to overcome the MBM limitations. The ReSOLV Model uses a custom blockchain transaction specification that includes additional fields tailored to the requirements of a flexible SLV schema. This provides the scope needed for the wide range of

users and license models in the modern technology environment. It can also provide several useful mechanisms using the blockchain as the basis for license validation, license upgrade, transfer of ownership, and even software integrity checking. A customised blockchain specification (Figure **??** on page ??) may include new blockchain fields, to improve SLV, and to prevent software piracy, through software integrity checks and protecting the software from reverse engineering and executable code modification. The fields in the customised blockchain are stored and encrypted with public/private keys. The software vendor uses the user's public key to encrypt the data being placed into the fields; the user subsequently applies the private key to decrypt the fields. The user can confirm transaction integrity signature with the vendor's public key.

While existing software validation uses digital signatures to verify downloadable software, and digital certificates to prevent Man-in-the-Middle attacks during the download process, the ReSOLV specification provides validation of installed software on the user's device on an ongoing basis, providing risk mitigation against malware code injection attacks. The ReSOLV Model provides:

1. That each software license is hard to copy because the license is represented by a transaction between vendor and user, is cryptographically verifiable, and is stored in the user's blockchain wallet. An adversary would require the password to the user's wallet in order to access the user's private key. Already, multi-factor authentication mechanisms are available to further enhance user wallet security. In addition, the license key does not have to be disclosed to the user, so the key cannot be copied. Every transaction is cryptographically secure and cannot be modified.
2. Software licenses are easily validated, through the blockchain "chain of title" and the data being held within the blockchain itself. Furthermore, having on-blockchain license keys allows the vendor to distribute keys for specific feature activations, and allows keys to be "re-cut" quickly and efficiently without the need for an intermediary.
3. Software licenses cannot be regenerated because the software application takes the license key directly from the blockchain, requiring the user's private key. Even if the key generator at the vendor is compromised, there is no way to get the license key onto the blockchain without the vendor's private key to sign the transaction.
4. No Man-in-the-Middle attack is possible, using the blockchain. An adversary cannot make use of any intercepted data in the blockchain, without the user's private key. Also, an adversary cannot redirect DNS or IP traffic to an adversary's custom server to provide software validation, without the vendor's private key.
5. Additionally, the peer-to-peer blockchain architecture means that no single central point of failure for SLV exists. Licensing validators can be run anywhere and, for example, could be run either commercially or on a not-for-profit basis. In particular, a vendor may run vendor-specific software to manage a license creation process and interaction with the blockchain, but does not need to maintain a dedicated license validation infrastructure with its associated overheads.

Therefore, the ReSOLV Model provides an opportunity for small to large software vendors to preserve software copyright and prevent software piracy, whilst having a flexible mechanism for software licensing. However, the ReSOLV model, unlike the MBM that can be run on an existing blockchain, requires a separate blockchain to be developed and maintained.

5.1. Requirements Specification

This section presents the requirements specification for the ReSOLV SLV ecosystem. The ReSOLV High Level Architecture (HLA) and ReSOLV Reference Architecture (RA) for functional requirements are introduced. Non-functional requirements, and public key cryptography, which lies at the heart of the core functions of ReSOLV. Laplante [58] states that the first requirement for development of a new system, is to develop a concise description of what it is supposed to do. Thus, the purpose of ReSOLV is to provide Software Piracy Prevention and Provenance (SPPP), using blockchain technology to validate user license entitlements, and to validate software integrity.

5.1.1. ReSOLV High Level Architecture

Three Primary Actors (PA) of the ReSOLV HLA are defined: Vendor (PA1), User (PA2), and ReSOLV Corp (PA3) (Figure 2 on the next page).

PA1: Vendor Three functional requirements are needed for the Vendor in the ReSOLV ecosystem: Provenance Agent/service that provides front-end interactions with blockchain Users and ReSOLV Corp; Software Encoding Service that provides back-end software encoding and licensing services; Vendor Wallet that provides two functions: a secure database function, and a mining function.

PA2: User The User actor requires two functions in the ReSOLV ecosystem: SmartWallet and the encoded software: SmartWallet is an application as a service that is part secure database, part cache, and part miner and provides the front-end interactions with the blockchain externally, as well as internal functions, servicing requests from software for license validation; Encoded Software is downloaded from the Vendor and requires authentication from the SmartWallet to validate user entitlements.

PA3: ReSOLV Corp ReSOLV Corp is the actor that has the primary off-blockchain role of supporting the on-going development of the ReSOLV ecosystem and providing a governance framework required to mitigate risks associated with the blockchain and increase value to the ecosystem actors. ReSOLV Corp provides the on-blockchain functions: Minting that includes the initial blockchain genesis block, minting of the initial tokens, and managing the token supply to all the Vendors, using its own wallet to store the tokens; blockchain ecosystem mining functions on an ongoing basis, as the most trusted actor on the blockchain.

The HLA may extend to include enterprises, where integration into Lightweight Directory Access Protocol (LDAP) or Active Directory (AD) services is required for corporate user identity validation, SaaS services that use blockchain authentication method over a centralised OAuth service, and supporting Internet actors for proof of identity, multiple signature support, and online wallets secured by hardware wallets such as the Trezor and LedgerWallet.

5.1.2. ReSOLV Reference Architecture

The RA illustrates human actors as the starting point to describe the primary processes and methods in the ReSOLV blockchain ecosystem (Figure 3 on page 14), related technology actors such as mining agents, wallet agents, encoding service, and provenance agents. Also illustrated are primary processes that are related to the human actors and group the ReSOLV Methods. The ReSOLV Methods themselves show the flows of data in the ReSOLV blockchain ecosystem and are labelled according the human actor.

Examples of RAs presently available include Wood [59] who published an RA for Ethereum proof-of-work and Alqassem and Svetinovic [60] who propose a Bitcoin RA focussing on the fundamental methods for the Bitcoin protocol.

Human Actors Three actors are shown: User (U), Vendor (V), and Corp (C). Corp is the physical entity that is responsible for the creation, development, governance, and ongoing operation of the ReSOLV blockchain ecosystem. The Vendor is the software company that develops applications to be protected with ReSOLV. User is the consumer that downloads, installs and runs the application from the Vendor. Each human actor has their own Wallet Agent to store tokens and private keys.

Primary Processes The Token processes (purple) are related to the creation and transfer of tokens that have not been distributed to a User with a license key (unserialised tokens). License processes (brown) are associated with the encoding of the Vendor application and license generation and issuance. Validation processes (green) provide user execution, authentication, and license validation. Events initiated by the actor are shown as dashed lines.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 March 2018 doi:10.20944/preprints201803.0217.v1

Peer-reviewed version available at Cryptography 2018, 2, 10; doi:10.3390/cryptography2020010

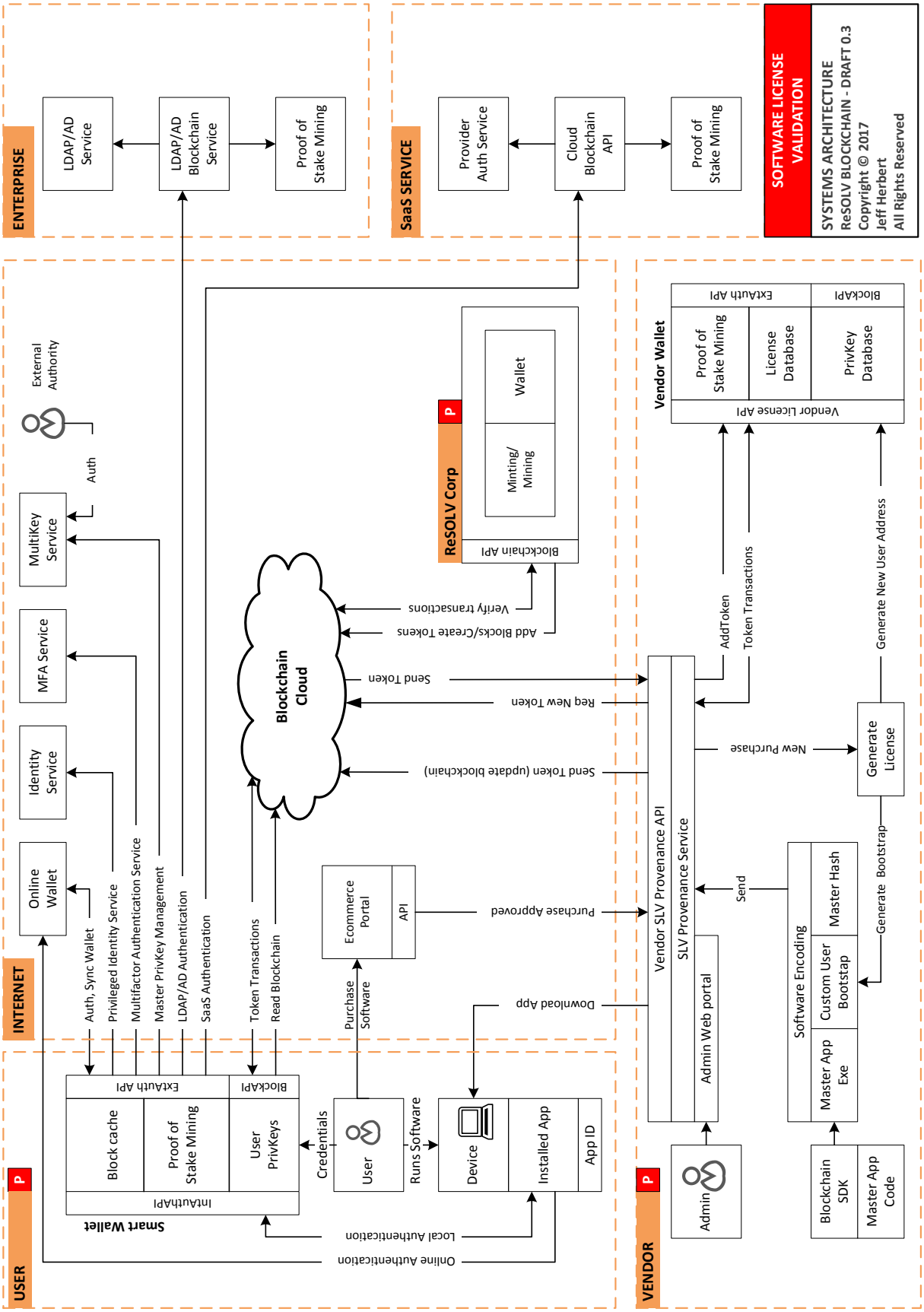

Figure 2. Proposed ReSOLV HLA

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 March 2018 doi:10.20944/preprints201803.0217.v1

Peer-reviewed version available at Cryptography 2018, 2, 10; doi:10.3390/cryptography2020010

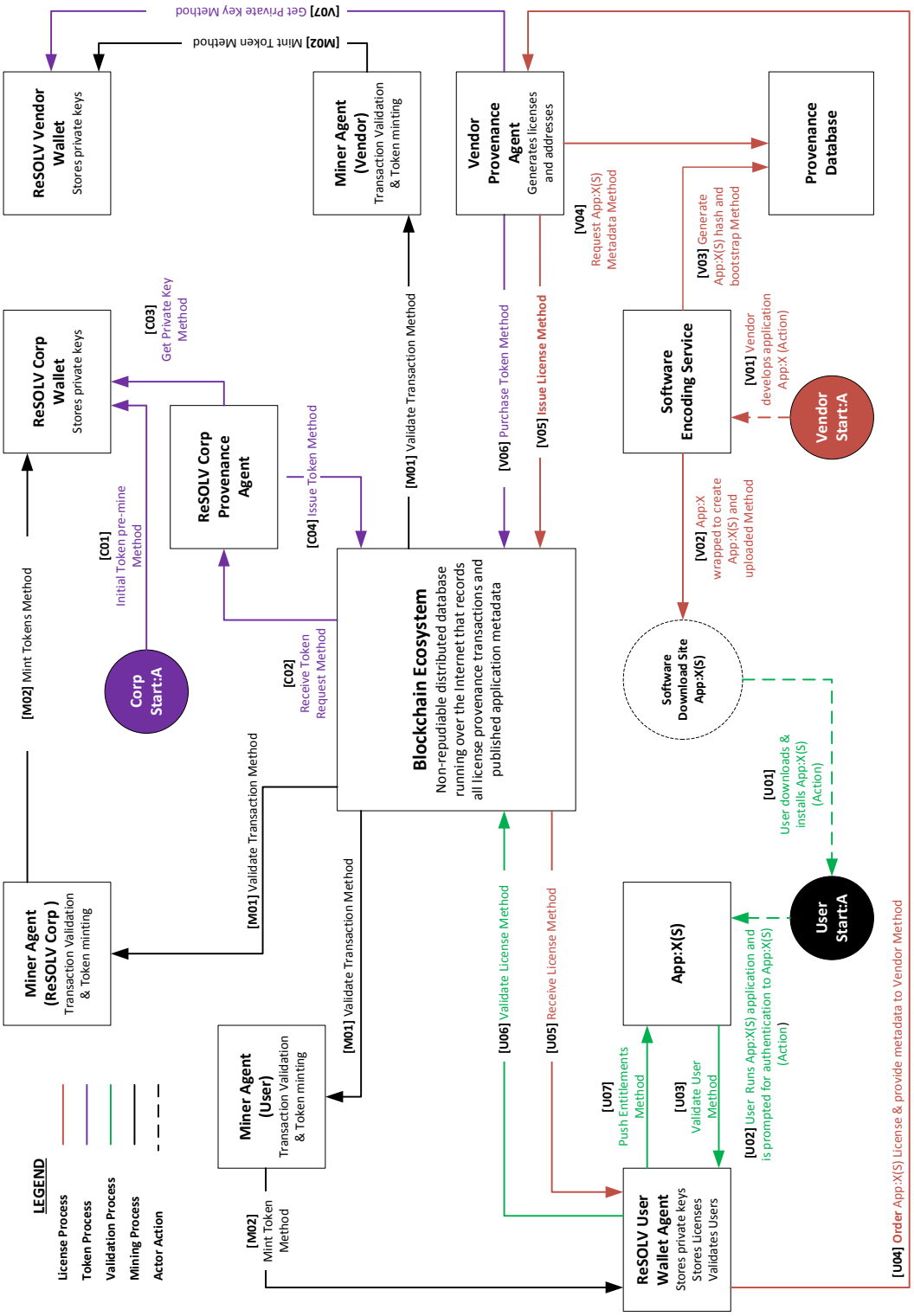

Figure 3. The ReSOLV RA

Technology Actors Include provenance, mining, and user wallet agents, and the software encoding service. The ReSOLV Corp provenance agent is responsible for token issuance to vendors, whilst the Vendor provenance agent is responsible for license issuance to the User. The Vendor also has a Software Encoding Service. The Mining agents participate in transaction validation and token minting processes, with newly minted tokens stored in each actor's wallet. This concept provides an economic incentive for the Vendor and User actors to create tokens for use within the ecosystem.

5.1.3. Non-functional Requirements

The non-functional requirements are considered from the perspectives of design/architecture, usability, and governance. Much discussion through various discussion forums has occurred regarding this topic and the consensus seems to be that these types of non-functional requirements need to be addressed during the design phase of the cryptocurrency or blockchain. For example, problems that have occurred when this was not done include the Bitcoin design stand-off between miners and developers, arguing over how to improve Bitcoin scalability that was not factored into the original Bitcoin design.

Design Requirements *Scalability* that includes transactions per second, block size, blockchain bloat, validation period, capability to handle micro-transactions, mining requirements [61,62,63]. *Robustness* of the blockchain ecosystem such that it is hard to disrupt through bugs, growth, delays in transaction validation, changing environment, or malicious attacks on the blockchain miners and protocols [64,65,66]. *Resiliency*, or the capability to provide operational continuity, or to recover to full operation with minimal effort [66,67]. *Data Integrity* that ensures the integrity of the blockchain through proven and trusted cryptographic methods [60,68]. *Security* such that transactions are secure, proofs are valid, and that double-spending does not occur [69,70,71].

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 can use the software without being connected to the Internet.

Governance Considerations Controls are present that *prevent hostile takeover* of the cryptocurrency or blockchain, for example by Miners that attempt to form a 51% collective mining power block [72, 73]. *Ownership controls* that ensure the funding entity retains ownership of the cryptocurrency or blockchain development once it has reached go-live [74,75]. *Stability* to ensure Miners are mining the blockchain using the latest stable release of mining software [75]. Economic design decisions for economic parameters to be incorporated into the cryptocurrency or blockchain to ensure *sustainable functionality* [75,76]. Policies are established to what *source code model* is used and where, private or open source [75,77].

5.2. Public Key Cryptography and Digital Signatures

Cryptocurrencies and blockchain technology use a range of cryptographic primitives to enable the decentralised trust and the immutability of the blockchain ledger. There are two primary cryptographic functions used by cryptocurrencies, Public key cryptography to store and spend money and cryptographic validation of transactions using digital signatures [78]. To briefly summarise what we should already know, given the use of these technologies has a long and well tested history [79,80]; Public key cryptography is a form of asymmetric cryptography, which uses key pairs (a public key and private key) to provide authentication and encryption of data (messages) that are exchanged between two parties over an insecure medium such as the Internet. The public key is a public identifier that can be freely shared with others, which is used by others to encrypt data being sent to the owner of the public key. The private key is kept secret and is used by the owner to decrypt received messages that have been encrypted with the public key. Unlike symmetric

key cryptography, public key cryptography does not require a secure exchange of secret keys and functions securely based on full disclosure of the public key. Public key cryptography relies on cryptographic algorithms. Cryptocurrencies primarily utilise Elliptic Curve Cryptography (ECC), which provides for fast, efficient computations with low key size and high security [59,81].

A second cryptographic function used by cryptocurrencies, digital signatures, also uses public key cryptography. Mathematically, digital signatures prove authenticity that a message received is really from the person claimed, and that the message contents have not been modified during transmission over an insecure medium [79]. The sender signs the message, where the output value of the signing function is called the digital signature.

To ensure transactional integrity and non-repudiation, cryptocurrencies use public-key cryptography to validate transactions between all participants, and digital signatures [38]. The cryptographic mechanisms used by cryptocurrencies provide for strong confidentiality, data integrity, and non-repudiation services that are in use by business, government, and military organisations, globally. In a cryptocurrency ecosystem, the public key can be considered as the participant's account number, whilst the private key represents the participant's ownership credentials. All participants have digital wallets that are used to store private keys, as well as digital signatures that represent cryptocurrency entitlements (bitcoins) that the participants own.

ReSOLV utilises public key cryptography and digital signatures for the transfer of tokens (the digital signature) between accounts (user wallets). It also utilises public key cryptography for encryption of Sidebar data. The Sidebar is created by an Issue License Method in which the Vendor provenance agent generates a license requested by a User with a unique blockchain address. The Sidebar contains a set of data fields with the license key, hash, and bootstrap, all encrypted with the User's public key, and signed with the Vendor private key. This process maintains the confidentiality and integrity of data residing on the blockchain. ReSOLV also utilises digital signatures to provide authenticity of the Sidebar itself, and places the Sidebar digital signature on the blockchain, so that any party can validate that the Sidebar is from the expected software vendor.

Each software item, shown as App:X in Figure 3, requires a new public-private key pair that will be used to store the digital signature for the license in the User Wallet Agent. The User Wallet Agent, as the master repository of all the private keys, will also need to be strongly protected itself, because if the wallet is compromised it will expose all the private keys.

In the ReSOLV Model, each license entitlement requires a unique address belonging to the organisation to be sent a Token, with the number of Tokens the organisation has available represented as addresses on the blockchain. That is, an organisation with 100 users would have 100 addresses in a wallet dedicated to the organisation. Licenses need to be allocated to users within the organisation and also be revoked. Furthermore, users need to authenticate using corporate login credentials, ideally using a single sign-on approach to access the license from the organisation's wallet. This requires some form of authentication service internally for the blockchain software license validation with capability to integrate into a service such as LDAP or AD for single sign on.

Additionally, the license validation blockchain method provides an opportunity to manage licenses on autonomously operated devices. As the Internet-of-Things grows and evolves, these connected devices will require mechanisms to auto-update and validate software. For example, the customer who owns 10,000 Internet-connected devices, but only pays software maintenance on 2000 of these devices will have license keys to update 2000 devices only. This capability is relatively straightforward to achieve in a peer-to-peer decentralised software license validation ecosystem but hard to manage using any other type of process. Perkins [82] states that the Identity of Things is a growing outcome of the Internet of Things. That is, a relationship exists between a device, data and a person, and that relationship needs to be defined and managed. The license validation method in the ReSOLV Model enables the definition of license entitlement, as a Token/source object, by providing the identities of associated people through the blockchain address and defining types of relationship between people and objects via transaction histories.

6. Discussion

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

- Establishing the scope of software piracy in respect to global impact, and expansion of platform piracy issues into the mobile computing world and gaming platforms.
- Identifying, defining and relating software piracy types and actors involved in soft-ware piracy, concluding by introducing a taxonomy of software piracy types
- Determining the methods used to protect software from piracy, and how threat actors defeat piracy protection mechanisms and overcome prevention methods.
- Relating how the threat actors interact and engage in the software piracy process, either deliberately or unwittingly, through the software piracy model.
- Identifying the points where software piracy can occur as shown by the Software Vulnerability Piracy Lifecycle.
- Identifying related software piracy issues such as enforcement, jurisdictional boundaries, and how pirated software is a well known vector for malware.

The attack surface shows that the path to software piracy prevention, and protection of software creator copyright, is challenging. Apart from Apple iOS, piracy is rife across all ecosystems and even Cloud Computing business models suffer considerable piracy due to credential sharing. The aggregated opportunity cost of software piracy across the various platforms is estimated to be USD132 billion annually, excluding economic costs from mobile and cloud platforms because there is little peer reviewed research available for these platforms. Thus there is motivation to address these issues.

To address the issues, a blockchain-based SLV method for software piracy prevention and provenance would need to be global in its reach and 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 have minimal infrastructure, place license authorisation into the hands of the software creator, prevent reverse engineering and cracking of software by threat actors or malware, uniquely bind user identity to software entitlements and only legitimate licenses issue, make credential sharing less attractive, and provide end user anonymity and data confidentiality by reducing the attack surface across all states of the software piracy vulnerability lifecycle.

Further, to meet software protection requirements across the attack surface and to overcome piracy issues, cryptocurrency and blockchain technologies are feasible. Blockchains are global, distributed, cost-effective, and customisable. Applications can be written to leverage existing cryptocurrencies, or can be designed or created as a new ecosystem dedicated to a specific use-case.

6.1. ReSOLV: a Native Blockchain Application

From a post-requirements specification and design perspective, Figure 4 on the next page illustrates ReSOLV from the client-side. This simplified diagram is taken from the ReSOLV RA and presents the core client-side innovation, a ReSOLV wallet agent and an application that has a wrapper applied that is executed ahead of the application to validate the user. Alice has a Wallet Agent that reads license information from the Blockchain Peer Pool, which it also stores in its local database (User and License Information Store). The database also holds user credentials and private keys related to the licenses, an essential requirement for the blockchain method to function. The application is wrapped with a ReSOLV executable, which first communicates with the Wallet Agent to validate the application's integrity using the stored hash of the application. It then requires Alice to enter her credentials to authenticate to her Wallet Agent, which, if Alice's credentials are correct, matches her entitlements and provides the application wrapper with the license and bootstrap.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 March 2018 doi:10.20944/preprints201803.0217.v1

Peer-reviewed version available at Cryptography 2018, 2, 10; doi:10.3390/cryptography2020010

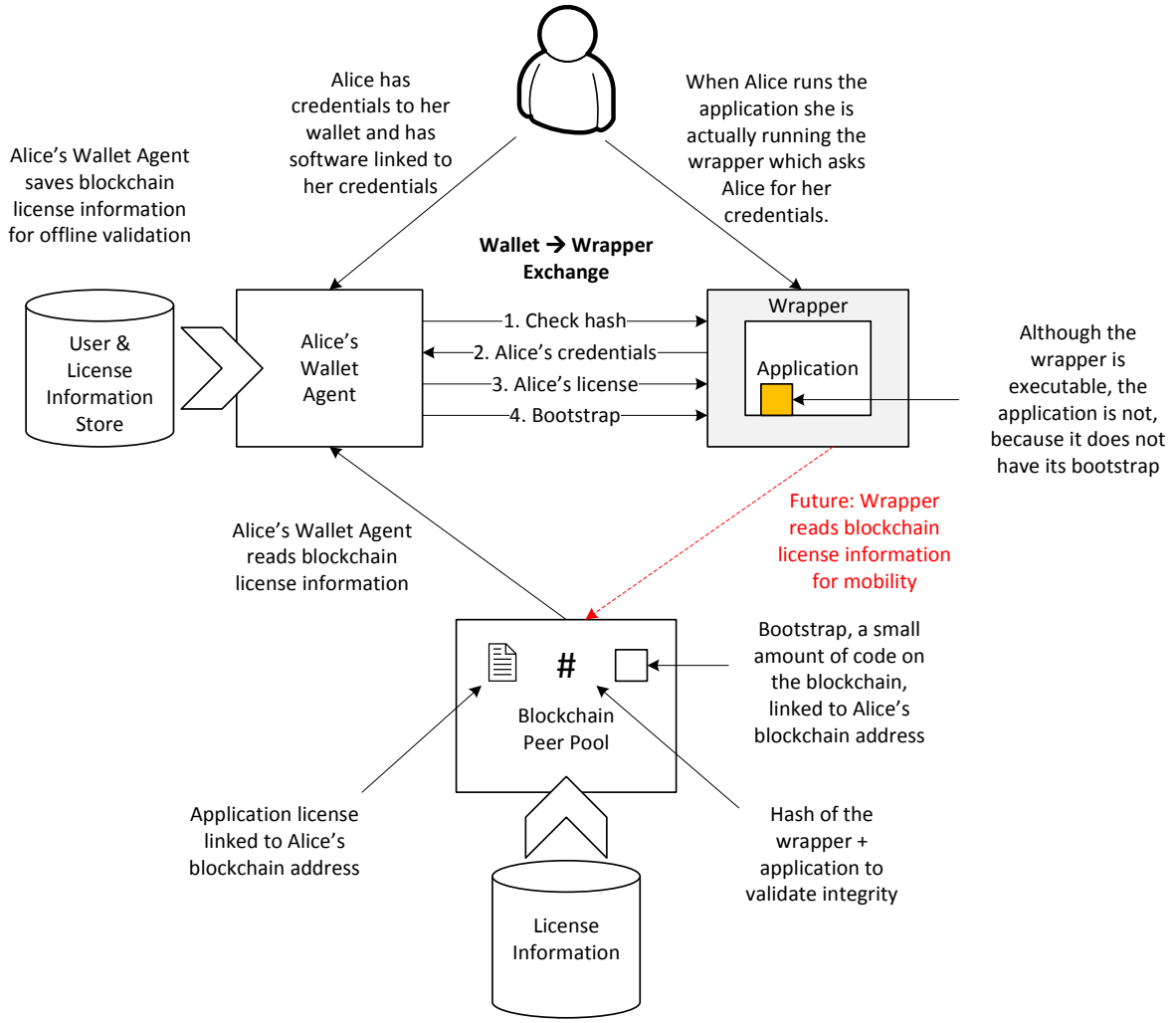

Figure 4. Simplified ReSOLV – Client-side

However, the HLA provided an enterprise perspective of ReSOLV and what the ecosystem may look like when integrated into third-party services for authorisation and support of enterprise directory environments, such as Microsoft's Active Directory. There will be future challenges, managing license entitlements for employees who will eventually depart the business that hired them. There will need to be a method for licenses to expire to allow for fluid situations, such as a dynamic workplace. ReSOLV already supports this through the software update mechanism.

Through the RE and FD design processes, it was established that Native Blockchain Applications (NBA) would suit ReSOLV model. The blockchain based method provides the core mechanisms for storage and distribution of the software licenses and data used to validate the application. This method eliminates man-in-the-middle attacks by using digital signatures from the vendor. Thus, licenses are protected from any copying process, as they are uniquely keyed to the Wallet Agent address and will not function anywhere else. Applications will not execute if any unauthorised modification of the application is undertaken. Applications will not execute without the bootstrap. If an attack attempts to capture the bootstrap in memory and insert it into the application, the application hash will not match, and the application will not execute in any case. Although a user can disclose their credentials and private keys to the Wallet Agent, the user is disincentivised to do so, because that would expose the license details for all their license entitlements.

On reviewing suitability of existing cryptocurrencies, such as Bitcoin and Ethereum, for SLV, it was found that these could provide the required technical functionality. However, the research contends that non-functional considerations, such as cost, and ecosystem considerations such as scaling, will create a persuasive weighting towards implementing ReSOLV as an NBA.

Although the design shows ReSOLV supports multiple platforms, User Wallet mobility needs to be considered. The User Wallet Agent contains the private keys that relate to each application license stored in the User Wallet database. For a user to be fully mobile, the User Wallet must also be mobile, to support the user authenticating from multiple devices. In addition, User Wallet security must be considered, as the User Wallet holds all the keys to the "crown jewels" and cannot be recreated if lost. In respect to cryptocurrencies, this problem has been partially addressed by wallet developers, such as Jaxx wallet or Exodus wallet. Known as "hot wallets", these applications have the capability to store private keys locally, and provides a 12-word backup phrase that can be entered into a wallet on another device, to allow a synchronisation of private keys. Whilst this is a useful backup mechanism that provides some mobility, it does not solve the user authentication problem if the User Wallet is not present. Other approaches include using a portable device to store the license private keys (for example, a special hardware wallet, such as Trezor, developing the application wrapper to read license information directly from the blockchain, or use cloud-based authentication services for users

While the ReSOLV client can detect unauthorised changes to a file with certainty, it cannot determine if the changes were malicious. ReSOLV has two processes supporting identification of unauthorised, modified software, that is, the software upgrade system ensures that every hash of every authorised application and its subsequent patches are recorded on the blockchain, and the ReSOLV User Wallet Agent acts as an independent authority, checking the hash of the application, based on the digitally signed hash on the blockchain (or in its database).

The ReSOLV RA (Figure 3 on page 14) and other design components show that data confidentiality and user privacy requirements are independent of the base cryptocurrency used. Therefore in respect of data confidentiality and privacy requirements, ReSOLV is cryptocurrency neutral.

7. Potential Issues

User authorisation is a critical process in ReSOLV method and it may be easy to attack and exploit. Using a cloud authentication service, such as OpenID or OAuth, or an authorisation service provider, such as Facebook or Google, may be required. Also, the User Wallet Agent is responsible for validating the vendor application. However, the User Wallet Agent itself should also undergo validation, to detect if the Agent itself has experienced an unauthorised modification. This process is not incorporated into the RE process, and protecting the User Wallet Agent will need to be considered, to prevent compromise. In addition, key management and security may become a challenge, with many public-private key pairs existing in ReSOLV, as each application has a new key pair for every transaction. Key loss or corruption will result in loss of licenses.

The ReSOLV transaction protocol is likely to have different requirements to the Bitcoin transaction protocol, and may require extensive adaptation of an existing cryptocurrency protocol. Examples include: Changes to the transaction output script, because Users are not spending ReSOLV tokens like bitcoins; the ReSOLV design may choose not to support the transfer of ReSOLV tokens once sent to the User Wallet Agent, as Vendors may not want software entitlement to be transferable; and, the Bitcoin scripting language may be unsuitable for ReSOLV, and a customised scripting language or Turing complete transaction protocol may be required for ReSOLV transactions.

8. Limitations

There are a number of limitations noted in this paper. The blockchain depends on having a significant number of actively participating miners in the cryptocurrency ecosystem to create blocks and validate transactions. 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.

There are currently no standards for cryptocurrency or blockchain technology available. However one of the authors, A. Litchfield, is now a member of the ISO Technical Committee tasked with developing standards.

Other issues may include the security threat model for loss of data or if the user loses control of a wallet and user credentials are exposed, or a vendor system is compromised. Multi-signature authorisation provides a potential solution to these issues however this may place an additional overhead on the software license validation method.

9. Conclusion

SLV has been a primary mechanism for the prevention of software piracy since the mid-1970's. SLV methods have evolved to include online license validation over the Internet, in addition to traditional paper based license keys provided with software packaging. Software pirates and hackers can reverse engineer and remove protection mechanisms whilst license keys are reused, duplicated or regenerated to provide a valid license key. We contend that software developers need a license validation method that provides a unique license key that cannot be reused or regenerated, and that associates the identity of the user with the license key and is cost effective.

We show that a customised cryptocurrency blockchain can be used to provide a decentralised peer-to-peer software license validation method that meets the requirements for SLV in a cost effective manner through the application of cryptocurrency theory. The blockchain provides the opportunity to include software integrity and protection mechanisms, thus adding value for software vendors and end users. The blockchain software license validation method provides a "hard-to-pirate" software platform especially useful for software vendors that are frequent victims of software piracy, such as those developing on Microsoft and Android platforms.

Author Contributions: Jeff Herbert conceived and developed most of the concepts under the research leadership of Alan Litchfield. Alan Litchfield provided systems analysis and design elements, guided the development process and constructed this paper.

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
HLA	High Level Architecture
IMEA	International Mobile Equipment Identity
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. *Computers & Security* **2003**, *22*, 316–321.
2. Morgan, M.J.; Ruskell, D.J. Software Piracy — The Problems. *Industrial Management \& Data Systems* **1987**, *87*, 8–12.
3. Sachan, A.; Emmanuel, S.; Kankanhalli, M. Efficient license validation in MPML DRM architecture. ACM. ACM, 2009, pp. 73–82.
4. Liu, Z.; Roychoudhury, A. Relating software validation to technology trends. *International Journal on Software Tools for Technology Transfer* **2012**, *14*, 631–638.
5. Herbert, J.; Litchfield, A. A Novel Method for Decentralised Peer - to - Peer Software License Validation Using Cryptocurrency Blockchain T echnology. 38th Australasian Computer Science Conference (ACSC 2015); , 2015; pp. 27–30.
6. Maude, T.; Maude, D. Hardware protection against software piracy. *ACM* **1984**, *27*, 950–959.
7. Mooers, C.N. Preventing Software Piracy. *Computer* **1977**, *29*, 29–30.
8. Suhler, P.A.; Bagherzadeh, N.; Malek, M.; Iscoe, N. Software Authorization Systems. *IEEE Software* **1986**, *3*, 34–41.
9. Im, J.H.; Van Epps, P.D. Software piracy and software security measures in business schools. *Information & Management* **1992**, *23*, 193–203.
10. Taylor, G.S.; Shim, J.P. A Comparative Examination of Attitudes Toward Software Piracy Among Business Professors and Executives. *Human Relations* **1993**, *46*, 419–433.
11. Conner, K.R. Software piracy: an analysis of protection strategies. *Management Science* **1991**, *37*, 125–139.
12. Katz, M.L. Systems competition and network effects. *The journal of economic perspectives* **1994**, *8*, 93–115.
13. Shy, O.; Thisse, J.F.J.F. A strategic approach to software protection. *Journal of economics & management strategy* **1999**, *8*, 163–190.
14. Darmon, E.; Rufini, A.; Torre, D. Back to software "profitable piracy": the role of information diffusion. *Economics Bulletin* **2009**, *29*, 543–553.
15. Business Software Alliance. The Compliance Gap. Technical report, Business Software Alliance, 2014.
16. Traphagan, M.; Griffith, A. Software Piracy and Global Competitiveness: Report on Global Software Piracy. *International Review of Law, Computers & Technology* **1998**, *12*, 431–451.

17. Andrés, A.R.; Goel, R.K. Does software piracy affect economic growth?: evidence across countries. *Journal of policy modeling* **2012**, *34*, 284–295.
18. Business Software Alliance. 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, BSA | The Software Alliance, BSA | The Software Alliance, 2015.
20. Khan, A.u.R.A.N.; Othman, M.; Ali, M.; Khan, A.u.R.A.N.; Madani, S.A. Pirax: framework for application piracy control in mobile cloud environment. *The Journal of Supercomputing* **2014**, *68*, 753–776.
21. Davies, C. http://goo.gl/BWhej1, accessed on 15 March, 2018.
22. Keyes, D. http://goo.gl/428ZmP, accessed on 15 March, 2018.
23. Smith, D. http://www.businessinsider.com.au/android-piracy-problem-2015-1, accessed on 15 March, 2018.
24. Computer Fraud and Security. Android marketplace hit by malware. *Computer Fraud & Security* **2011**, *2011*, 3.
25. Sigi, G. Exploring the supply of pirate software for mobile devices: An analysis of software types and piracy groups. *Information Management & Computer Security & Computer Security* **2010**, *18*, 204–225.
26. Han, K.; Shon, T. Software authority transition through multiple distributors. *The Scientific World Journal* **2014**, *2014*, 295789.
27. BBC Technology desk. http://www.bbc.com/news/technology-34338362, accessed on 15 March, 2018.
28. Davies, C. https://www.cnet.com/news/virus-scanners-filled-with-malware-are-flooding-app-stores/, accessed on 15 March, 2018.
29. Palmer, B.P. Anonymously Establishing Digital Provenance in Reseller Chains. phdthesis, Victoria University Of Wellington, Wellington, 2014.
30. Xiaosong, L.; Kai, H. Collusive Piracy Prevention in P2P Content Delivery Networks. *Computers, IEEE Transactions on* **2009**, *58*, 970–983.
31. ISO/IEC. https://www.iso.org/obp/ui/#iso:std:iso-iec:19770:-1:ed-3:v1:en, accessed on 15 March, 2018.
32. Veerubhotla, R.S.; Saxena, A. A DRM Framework Towards Preventing Digital Piracy. 2011 7th International Conference on Information Assurance and Security (IAS); , 2011; pp. 1–6.
33. Fortin, C. Master Bitcoin - The Proof of Ownership.
34. Lebo, A. https://github.com/fisher-lebo/dissent, accessed on 15 March, 2018.
35. Ford, P. Marginally useful: Bitcoin itself may not flourish as a currency, but the underlying technology is beginning to suggest valuable new applications. *MIT Technology Review* **2014**, *117*, 80.
36. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system.
37. Brikman, Y. http://brikis98.blogspot.fr/2014/04/bitcoin-by-analogy.html, accessed on 15 March, 2018.
38. Peteanu, R. http://bitcoinmagazine.com/11599/fraud-detection-world-bitcoin/, accessed on 15 March, 2018.
39. Bradbury, D. The problem with Bitcoin. *Computer Fraud & Security* **2013**, *2013*, 5.
40. Courtois, N.T. Crypto Currencies And Bitcoin.
41. Hanley, B.P. The False Premises and Promises of Bitcoin.
42. Plassaras, N.A. Regulating Digital Currencies: Bringing Bitcoin within the Reach of the IMF. *Chicago Journal of International Law* **2013**, *14*, 377.
43. Bass, E.; Bault, T.; Baum, A.; Channell, J.; Englander, S.; Ghose, R.; Ho, S.; Hopkins, D.; Juvekar, P.; May, M.; McDonald, G.; Michaeli, I.; Pritchard, W.; Sasaki, T.; Singlehurst, T.; Thein, T.; Wong, K. https://www.citivelocity.com/citigps/ReportSeries.action?recordId=25, accessed on 15 March, 2018.
44. Duivestein, S.; Savalle, P. http://thenextweb.com/insider/2014/02/15/bitcoin-platform-currency/1/, accessed on 15 March, 2018.
45. Prisco, G. The New Stellar Consensus Protocol Could Permit Faster and Cheaper Transactions. *Bitcoin Magazine* **2015**.
46. Buterin, V. Mastercoin: A Second-Generation Protocol on the Bitcoin Blockchain. *Bitcoin Magazine* **2013**.
47. Halford, R. http://www.gridcoin.us/images/gridcoin-white-paper.pdf, accessed on 15 March, 2018.
48. King, S.; Nadal, S. PPCoin: peer-to-peer crypto-currency with proof-of-stake.
49. Hochstein, M. Why bitcoin matters for bankers. *American Banker* **2014**, *March 2014*.
50. McCook, H. An Order-of-Magnitude Estimate of the Relative Sustainability of the Bitcoin Network.

51. Baran, P. *On Distributed Communications: I. Introduction to Distributed Communications Networks*; RAND Corporation: Santa Monica, CA, 1964.
52. Buterin, V. Bootstrapping A Decentralized Autonomous Corporation. *Bitcoin Magazine* **2013**.
53. Larimer, D. Transactions as Proof-of-Stake.
54. Wood, G. Ethereum: A Secure Decentralised Generalised Transaction Ledger Final Draft - Under Review.
55. Community, N.X.T. https://wiki.nxtcrypto.org/wiki/Whitepaper:Nxt, accessed on 15 March, 2018.
56. Bentov, I.; Gabizon, A.; Mizrahi, A. Cryptocurrencies without Proof of Work. *CoRR* **2014**.
57. Sommerville, I. *Software Engineering*, ninth edition ed.; Addison Wesley: Boston, MA, 2010.
58. Laplante, P. *Requirements Engineering for Software and Systems*; CRC/Taylor & Francis: Boca Raton, 2014.
59. Wood, G. http://gavwood.com/Paper.pdf?TB_iframe=true&width=288&height=432, accessed on 1 August, 2016.
60. Alqassem, I.; Svetinovic, D. Towards reference architecture for cryptocurrencies: Bitcoin architectural analysis. Proceedings - 2014 IEEE International Conference on Internet of Things, iThings 2014, 2014 IEEE International Conference on Green Computing and Communications, GreenCom 2014 and 2014 IEEE International Conference on Cyber-Physical-Social Computing, CPS 20, 2015, pp. 436–443.
61. Buterin, V.; Wood, G.; Zamfir, V.; Coleman, J. Notes on Scalable Blockchain Protocols, 2015.
62. Danezis, G.; Meiklejohn, S. Centrally Banked Cryptocurrencies.
63. Oberhauser, A. Decentralized Public Ledger as Enabler for the Gift Economy at Scale.
64. Mazières, D. https://www.stellar.org/papers/stellar-consensus-protocol.pdf, accessed on 22 March , 2018.
65. 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.
66. Glaser, Florian (Goethe University, F.; Bezzenberger, Luis (Goethe University, F. Beyond Cryptocurrencies - A Taxonomy Of Decentralized Consensus. 23rd European Conference on Information Systems (ECIS 2015), 2015, pp. 1–18.
67. Bott, J.; Milkau, U. Towards a Framework for the Evaluation and Design of Distributed Ledger Technologies in Banking and Payments. *Journal of Payments Strategy & Systems* **2016**, *10*, 153–171.
68. European Central Bank. *Virtual Currency Schemes (2015)*; European Central Bank: Frankfurt, Germany, 2015; p. 34.
69. Miers, I.; Garman, C.; Green, M.; Rubin, A.D. Zerocoin: Anonymous Distributed E-Cash from Bitcoin. 2013 IEEE Symposium on Security and Privacy. IEEE, 2013, pp. 397–411.
70. 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. *Communications of the ACM* **2013**, *59*, 127–140.
71. Xiaochao, Q. Build a Compact Cryptocurrency System Purely Based on PoS.
72. Buterin, V. Mining Pool Centralization At Crisis Levels. *Bitcoin Magazine* **2014**.
73. Ali, M.; Nelson, J.; Shea, R.; Freedman, M. Blockstack : A Global Naming and Storage System Secured by Blockchains. USENIX Annual Technical Conference; , 2016; pp. 181–194.
74. NXT Community. https://wiki.nxtcrypto.org/wiki/Whitepaper:Nxt, accessed on 15 March, 2018.
75. Millar, J. https://entethalliance.atlassian.net/wiki/download/attachments/37151/EntEthVision-v3.1-24February2017.pdf?version=1&modificationDate=1488946292762&cacheVersion=1&api=v2, accessed on 1 August, 2017.
76. Fry, J.; Cheah, E.T. Negative bubbles and shocks in cryptocurrency markets. *International Review of Financial Analysis* **2016**, *47*, 343–352.
77. Martin, K. Regulating Code: Good Governance and Better Regulation in the Information Age, by Ian Brown and Christopher Marsden. Cambridge. *Business Ethics Quarterly*, *24*, 624–627. book review.
78. Böhme, R.; Christin, N.; Edelman, B.; Moore, T. Bitcoin: Economics, Technology, and Governance. *Journal of Economic Perspectives* **2015**, *29*, 213–238.
79. Rivest, R.L.; Shamir, A.; Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* **1978**, *21*, 120–126.
80. Boneh, D.; Sahai, A.; Waters, B. Functional Encryption : A New Vision for Public-Key Cryptography. *Communications of the ACM* **2012**, *55*, 56–64.

81. Singh, S.; Jeong, Y.S.; Park, J. A survey on cloud computing security: Issues, threats, and solutions. *Journal of Network and Computer Applications* **2016**, *75*, 200–222.
82. Perkins, E. http://blogs.gartner.com/earl-perkins/2014/08/04/the-identity-of-things-for-the-internet-of-things/, accessed on 15 March, 2018.