A Novel Method for Decentralised Peer-to-Peer Software License Validation Using Cryptocurrency Blockchain Technology

Jeff Herbert
School of Computer and Mathematical Sciences
Auckland University of Technology
Auckland, New Zealand
Jeff.herbert@gmail.com

Alan Litchfield
Service and Cloud Computing Research Lab
Auckland University of Technology
Auckland, New Zealand
Alan.litchfield@aut.ac.nz

Abstract

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. This paper presents a novel method for decentralised peer-to-peer software license validation using cryptocurrency blockchain technology to ameliorate software piracy, and to provide a mechanism for all software developers to protect their copyrighted works.

Keywords: Cryptocurrency, blockchain, software, license, validation

1 Introduction

Methods to maintain control of copyrighted software have fallen into three main categories: software activation using a paper based key code, software license validation through an online registration (Peyravian, Roginsky, & Zunic, 2003) and hardware devices (Morgan & Ruskell, 1987). Smaller vendors most often implement software validation in the form of an activation key, whilst global vendors such as Microsoft and Adobe use proprietary centralised software license validation services using the Internet as the primary medium.

Software license validation is growing in complexity due to a combination of technological and economic developments. Commercial models for software sales and distribution have become more complex, with multiple parties existing in the supply chain including software owners, multiple levels of distributors and customers (Sachan, Emmanuel, & Kankanhalli, 2009). Similarly, software is becoming more complex as the scope of use increases (Liu & Roychoudhury, 2012).

This paper proposes the utilisation of a cryptocurrency blockchain similar to Bitcoin, to provide a method for decentralised, peer-to-peer, publicly auditable software license validation that could be used by anyone from an independent software writer to a large software vendor. We provide an overview of cryptocurrency blockchain functions and discuss the benefit of a decentralised peer-to-peer architecture. We then proceed to outline a construct of a transaction message and processes for blockchain-based software license validation, and explore future possibilities and issues.

2 Software license validation

2.1 Software piracy

The Business Software Alliance (BSA) defines software piracy as the unauthorised copying or distribution of copyright software, including downloading, sharing, selling, or installing multiple copies of licensed software. The Internet has provided a convenient medium for software piracy, enabling participants to easily download copyright software, and globalising software piracy by operating in difficult legal 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, and even subscription based models such as cloud computing are not expected to provide a significant impact on reducing software piracy with 52% of online credentials being shared (Business Software Alliance, 2014).

Methods to protect software creators' copyright have been in place since the early 1980's with a variety of methods proposed and implemented. Suhler, Bagherzadeh, Malek, and Iscoe (1986) suggested that to be successful, software authorisation (validation) needed to be inexpensive, compatible with other systems, and easy to implement. Similarly, Morgan and Ruskell (1987) found various practical measures to deter or prevent unauthorised copying, however the feasibility of these measures depend on various factors such as cost of the measure versus value of the software. Three primary methods for software license authorisation are considered: copy protection, software validation using a distributed paper-based key and hardware-based keys. In these nascent stages of computing, the more effective methods of encryption and validation were limited due to the relatively high cost of hardware devices, limited computing power for encryption methods and no form of easy distribution medium for software license validation. Software copy protection was primarily restricted to

alteration of disk sectors to prevent copying, which was easily defeated with software tools and license keys that were distributed with the software media and easily duplicated. These did little to resolve the issue of software piracy.

With the advent of the Internet, new methods became possible. Peyravian et al. (2003) proposed a new client-server software license validation method using the Internet with a central database for software license validation and detection of hardware platform characteristics that the software is installed on. Using this method, vendors need to manage end user information and need an online validation process for activation of the software after installation. Larger software corporations such as Microsoft and Adobe have adopted the principles of this method. However, online validation requires a significant overhead in management of customers, maintaining security of personal information and yet the validation method is defeatable through easy means, such as redirecting DNS to fake authentication servers, code modification to remove software license validation subroutines, or could be still circumvented through duplication of keys as many license models and license keys support installation on multiple devices.

2.2 Methods to comply with software licensing

Software license validation is growing in complexity due to a combination of technological and economic developments. Commercial models for software sales and distribution have become more complex, with multiple parties existing in the supply chain including software owners, multiple levels of distributors and customers (Sachan et al., 2009). Similarly, complying with software licensing is becoming more complex as the scope of software use increases, such as feature specific enablement keys for software packages, geographical diversity of where software is employed, size of customer organisations, shifts to software as a service models and an increasing use of embedded systems that leads to Internet of Things (Liu & Roychoudhury, 2012).

The need for Software Asset Management (SAM) has developed as organisations and users attempt to comply with complex software licensing requirements. Organisations can choose to manage their software licenses through established SAM processes and standards such as ISO/IEC19770, which provides guidance for organisations to manage software, including assessment of conformity, software identification, and software entitlements (ISO/IEC, 2012). The BSA has established Verafirm to assist organisations with the management of their software licensing, that provides SAM tools and solutions for SME's and enterprises.

3 Requirements for a software license validation method

As Suhler, Bagherzadeh, Malek, and Iscoe (1986) and Morgan and Ruskell (1987) stress, 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 require several premises to be met:

1) The license mechanism needs to be hard to copy
2) Rights to software licenses need to be easily validated
3) Software licenses cannot be repeatedly generated
4) Validation needs to protect from Man-in-the-Middle attacks

Therefore we need a mechanism that can generate unique values that can't be regenerated but can be easily verified against the source engine at any time. Cryptocurrencies such as Bitcoin already provide the essential building blocks we need for software license validation. Bitcoins are represented as cryptographically validated digital signatures and as such, unfeasible to copy, whilst the decentralised transaction feature prevents double spending of the bitcoin, ensuring 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. Hence, to meet the premises listed for software license validation, we propose a cryptocurrency blockchain to create a novel method for software license validation mechanism. The following section introduces the cryptocurrency blockchain and applies the blockchain concepts to software license validation.

4 Cryptocurrencies and the blockchain

Cryptocurrencies are a new form of virtual currency, first introduced with creation of Bitcoin, developed by Satoshi Nakamoto (2008). 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 including cryptographic validation for all transactions, decentralised money, mint and transactions, all stored on public ledgers within a quasi-anonymous framework (Brikman, 2014). Cryptocurrencies use public-key cryptography to validate transactions between all participants, and digital signatures to ensure transactional integrity and non-repudiation (Peteanu, 2014). The cryptographic mechanisms used by cryptocurrencies provide strong confidentiality, data integrity and non-repudiation services (NIST, 2001) and 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 the private keys, as well as the digital signatures that represent the cryptocurrency entitlements (coins) that the participants own. Wallets can be stored privately, or online on websites or exchanges depending on the requirements of the participant.

Cryptocurrencies as a currency and monetary system have yet to prove their robustness in both a technological and economic context, needing to be resilient to threats

and attacks as well as being a stable and liquid currency. However, the underlying feature of interest in respect to the cryptocurrency architecture is the blockchain, which is becoming the focal point of development of new cryptocurrency based applications as developers seek to use cryptocurrencies in more practical applications.

4.1 Transactions

Transactions are defined as a message between participants, and consists of 3 segments:

1) Signature: the originator's digital signature signed with the originator's private key so that other Bitcoin nodes can verify the message really came from the originating participant.
2) Inputs: this is a list of the signatures of transactions already in the ledger where the originator was the recipient of bitcoins. These are the funds the originator is using in the transaction.
3) Outputs: this is a list of how the funds in the inputs should be distributed. All the funds in the inputs must be redistributed in the outputs, so the originator will pay the recipient the required amount and return the remainder as change.

A transaction must have exactly the same number of bitcoins in the inputs and outputs. Hence if user U1 has 10 bitcoins, and wants to send 2 bitcoins to user U2, the transaction will result in U1 receiving 8 bitcoins, and U2 receiving 2 bitcoins. This can be shown as follows:

U1.input(U1, 10)
U1.send(U2, 2)
U1.send(U1, 8)

The recipient is identified through their public key, so cryptocurrency transactions can be traced throughout the blockchain, to the beginning of the creation of the cryptocurrency. This forms the mechanism for checking the ownership of cryptocurrency bitcoins. Publicly verifiable transactions by any node avoids double spending and provides a high degree of certainty to the participants of the cryptocurrency ecosystem.

4.2 The blockchain

The blockchain consists of a series of blocks where each block contains:

1) transactions or messages sent between users;
2) a unique digest created when the new block is discovered, called "Proof-of-work";
3) the previous reference to the digest of the previous block.

Figure 1 illustrates how each block has a proof-of-work of the previous block, forming the blockchain. Unverified transactions are placed in an unverified transaction bucket, and will be inserted into the next block once it is created.

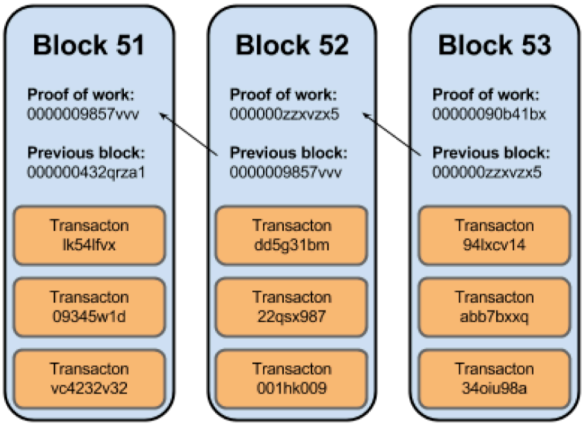

Figure 1: Example of the blockchain (Brikman, 2014)

4.3 Cryptocurrency and blockchain economics

The concept of a cryptocurrency is to overcome the necessity of a centralised "trusted authority" (Nakamoto, 2008) and thus remove or significantly reduce transaction fees associated with transactions such as those incurred with commercial banking transactions.

A cryptocurrency, as a peer-to-peer decentralised technology, relies on a network of low cost computers running software that performs the primary functions of the cryptocurrency. The computers running this software are known as miners, who 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 through creation of bitcoins, and receiving a small transaction fee for their part in validating bitcoin transactions. A cryptocurrency ecosystem requires a significant number of miners to manage the integrity of the blockchain and prevent double spending of bitcoins. However, depending on the implementation of the cryptocurrency ecosystem, miners may not receive transaction fees. For example, Ripple and Gridcoin cryptocurrency participants run the transaction validation software on a voluntary non-profit basis, offering their existing compute and storage resources to run the mining software.

Most cryptocurrency ecosystems have a fixed number of bitcoins that can be created, creating a deflationary economic model due to the finite number of bitcoins as bitcoin value inherently rises due to the limited supply of bitcoins. Bitcoins can also only be created at a certain rate, determined mathematically by the cryptocurrency ecosystem to prevent an oversupply of bitcoins. However, some cryptocurrencies such as Peercoin are established on an inflationary economic model, with an unlimited supply of bitcoins.

These approaches lead to cost effective cryptocurrency and blockchain ecosystems through lower transaction fees (Hochstein, 2014) for the cryptocurrency as a financial instrument. Furthermore cryptocurrencies are found as being considerably lower cost than fiat currencies when comparing economic, environmental and socioeconomic costs (McCook, 2014).

In the next section the characteristics of the blockchain that will help provide a decentralised software validation method are described.

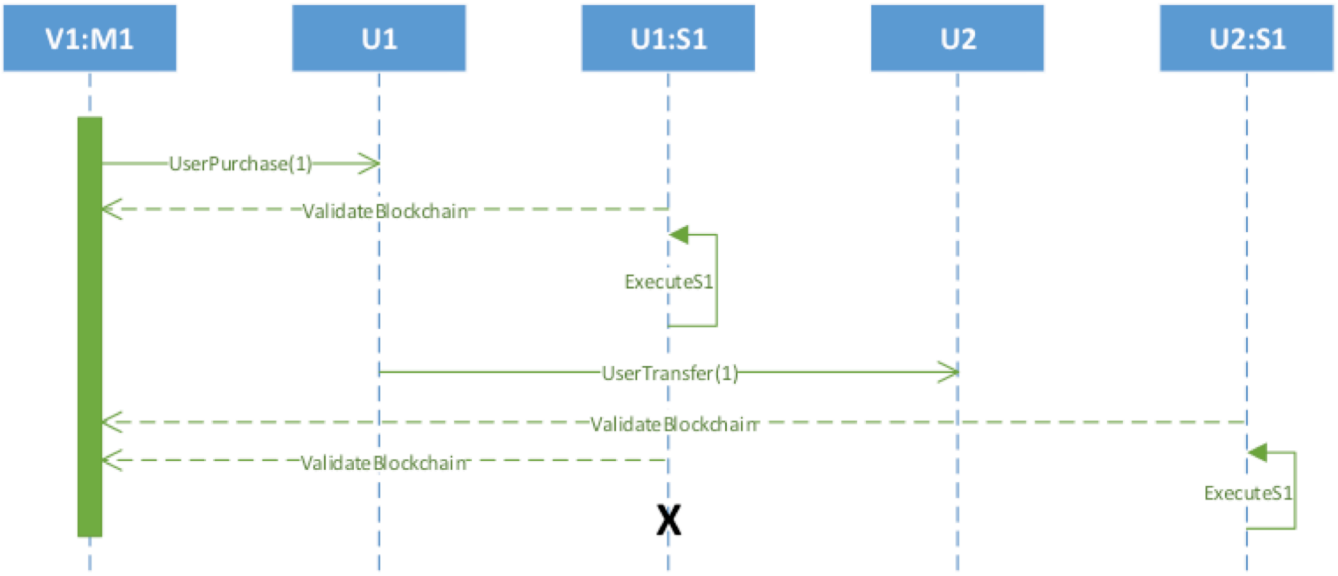

Figure 2: Master Bitcoin Model transfer of ownership sequence example

5 Decentralised software license validation

For the purposes of discussing decentralised software license validation, the term *bitcoin* is used generically as a descriptor for a virtual coin from an existing cryptocurrency. Bitcoins are actually digital signatures that are created and stored in user wallets, and have a full publicly verifiable transaction history through the blockchain transaction history. The characteristics of the blockchain can be utilised to provide a record of all software licenses owned by an end user. Through the decentralised peer-to-peer blockchain architecture, any software developer or vendor can allocate licenses to users easily and cost effectively. The principle of decentralised software license validation is to use bitcoins held by the owner to represent entitlement to software.

Two primary methods to utilise a blockchain for software license validation are the "Master Bitcoin Model" and the "Bespoke Model", discussed in the following sections.

5.1 Master Bitcoin Model

The Master Bitcoin Model is a basic form of software license validation proposed by Fortin (2011) and implemented by Lebo (2014) in a proof of concept project called "dissent", using Namecoin as the underlying blockchain. In this model, the vendor address/bitcoin combination represents license ownership, and if the user has a transaction showing the bitcoin originated from a specific vendor address, the user is considered to have ownership of the software. This concept is demonstrated in the following example.

The entities:

Vendor1 (V1): owns the Software application S1
Software1 (S1): the particular Software application
MasterAddress1 (M1): the address representing S1
UserAddress1 (U1): the end user address for the wallet that holds the bitcoin indicating software entitlement

1) V1 creates the M1 "MasterAddress1" on the blockchain, representing a particular Software application.
2) V1 then adds some bitcoins to M1, loading it with some bitcoins that when transferred will represent entitlement to the Software application.
3) The end user purchases the Software application through a non-cryptocurrency transaction.
4) V1 transfers a Master bitcoin from the M1 address to the U1 address. The transaction itself confers the ownership of the bitcoin, and the end user now has the bitcoin from M1, the Master bitcoin, in the user's associated wallet. Hence, the user's ownership of a bitcoin from M1 confers entitlement to the Software application, and is a transaction publically verifiable on the blockchain.
5) The Software application then validates that U1 has received a transaction from M1, and is the last transaction in the chain of transactions.

The sequence of transactions can be shown as follows:

```
V1.create(M1)
V1.send(M1, 100)        'V1 adds 100 bitcoins to M1
Software purchase
M1.send(U1, 1)
S1.validate(U1)
```

Ownership of the Master bitcoin can be transferred as shown in Figure 2, so that the software vendor can be guaranteed only a single user is using the software though checking the blockchain "chain of title" for the Master bitcoin originating address. Hence, U1 can now transfer ownership to a new party, U2. Again, the transaction itself confers the ownership, and any entity can verify the chain of transactions from U2, to U1, and back to M1 to

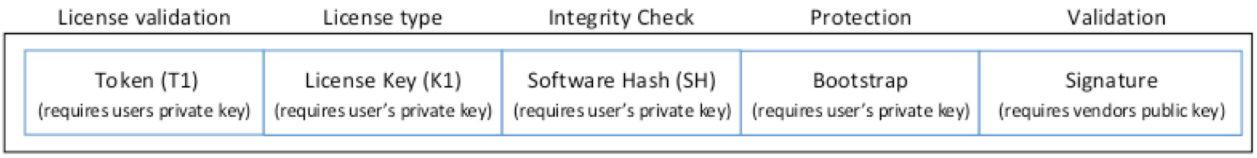

Figure 3: Customised blockchain specification for License Validation

confirm that current ownership is held by U2, who will have the last transaction in the chain of transactions.

M1.send(U1, 1), U1.send(U2, 1)

Since bitcoins do not have serial numbers, once a non-Master bitcoin is combined with a Master bitcoin, the originator of each specific bitcoin cannot be identified because in reality, they are simply digital signatures that have been combined to form a new digital signature. However, in the Master Bitcoin Model, the value of the Master bitcoin is not important, only the fact that there is a transaction history from the originating Master bitcoin. Fortin (2011) proposes that if a Master bitcoin is combined with a non-Master bitcoin, the biggest recipient is the one that holds the Master Bitcoin, or whoever has the lowest address (alphanumerically) has precedence for ownership of the Master bitcoin. This property establishes non-divisible ownership of the Master bitcoin allowing ownership to be transferred.

In summary, using a unique blockchain address to represent a particular software application, the Master Bitcoin Model can be used to provide non-repudiable proof of ownership of a bitcoin that originated from a specific address, thereby conferring the entitlement of the software license to the user. However, the software application will need to have the capability to read the blockchain to establish the chain of title to the user.

5.2 Bespoke Model

As mentioned earlier, most cryptocurrencies are designed with a currency in mind and so they create virtual coins represented as digital signature that are stored in users' wallets and have a full publically verifiable transaction history that is stored on the blockchain. For the purposes of discussing the Bespoke Model, we define a Token as a digital signature that represents entitlement to a specific software application, rather than a bitcoin, because the license validation model is not using digital signatures to represent a virtual currency. A user address that holds a particular Token from a specific vendor address is entitled to the software license, and therefore is entitled to use the software. Hence, the vendor/token combination represents the entitlement for use of the software.

Blockchain specifications vary from cryptocurrency to cryptocurrency, and as such, cryptocurrencies can be architected with unique characteristics to meet purpose specific applications. The Bespoke Model uses a custom blockchain transaction specification that includes additional fields tailored to the requirements of a flexible software license validation schema. This would provide the scope needed for the wide range of users and license models in the modern technology environment. We 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, as shown in Figure 2, could include new blockchain fields to improve software license validation and prevent software piracy through software integrity checks and protecting the software from reverse engineering and executable code modification.

These fields are all stored on the blockchain as data, encrypted using the in-built cryptocurrency public/private key mechanisms. In principle, the software vendor utilises the user's public key to encrypt the data being placed into the fields, with the user's private key required to decrypt the fields. The user can confirm the transaction integrity signature with the vendor's public key.

The custom fields outlined in the proposed specification are described as follows.

The Token is used for standard license validation mechanisms where the ownership of the Token demonstrates entitlement. The Token can be used for software license validation operations such as for software upgrades, or to provide a unique attribute to the transaction, such as "first 100 purchasers" that may have collectible value in the future.

The License Key provides advantages over the Master Bitcoin Model because many software applications have specific features within the application that are activated on a per feature basis. Having the License Key securely held on the blockchain means software vendors can easily enable "feature activation", and have flexibility with software application licensing models, where users could rent software use for a small periods of time, rather than purchasing or renting use on a month basis.

Similarly, the vendor can place a software hash of the application on the blockchain. A bootstrap loader or the software itself can read the hash and check the software version. This hash can be updated with every new patch, plus minor or major releases of the software. This could protect software from malware infection or some forms of reverse engineering.

Additional protection could be provided through a bootstrap loader, which is a portion of executable code that is used to pre-execute the software application or to be used as an integral part of application execution. The purpose of this is to further prevent reverse engineering of the software application. At some stage the unencrypted bootstrap code will be executed and stored in memory, and thus susceptible to interception by reverse engineering. This bootstrap code can change with every patch, and minor and major release, making reverse engineering a constant effort.

The signature field is a possible additional field that can be used by the vendor to sign the entire transaction contents using the private/public key pair of the software MasterAddress.

Existing software validation uses digital signatures to verify downloadable software and digital certificates to

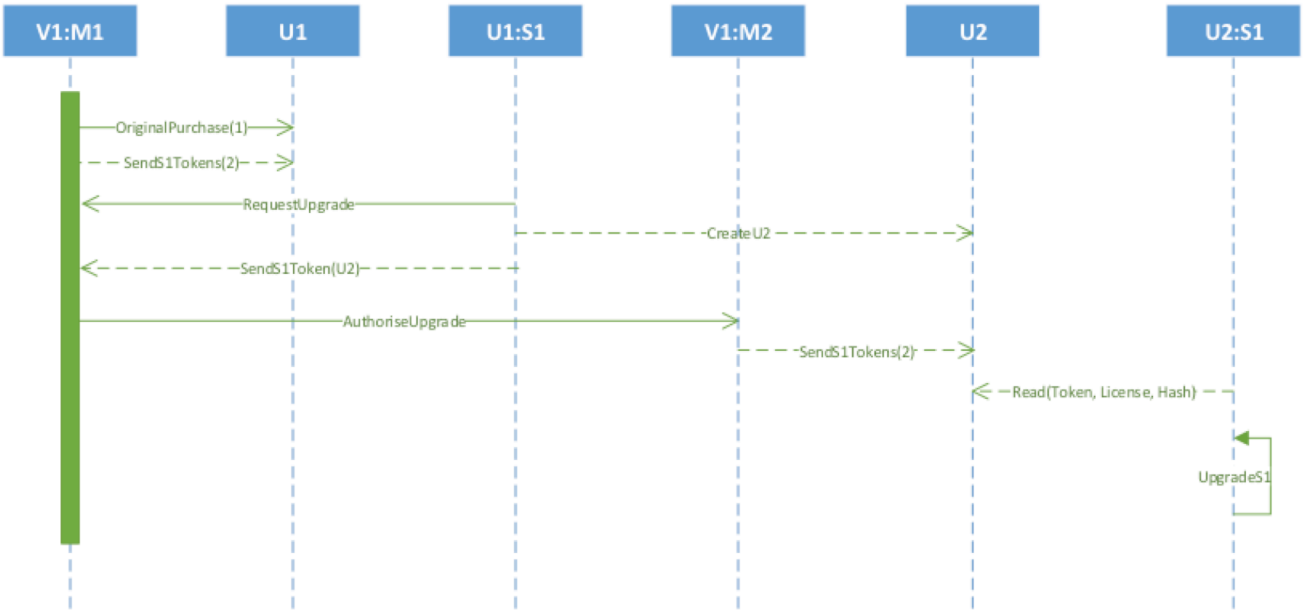

Figure 4: Bespoke blockchain software upgrade sequence example

prevent Man-in-the-Middle attacks during the download process. However the proposed custom specification provides validation of installed software on the user's device on an ongoing basis, providing risk mitigation against malware code injection attacks.

Exploring these concepts further, we look at the Token feature. As already mentioned, the Token is used to confer ownership, however in comparison to the Master Bitcoin Model, the Bespoke Model presents significantly more opportunities to use the Token for software license validation purposes. In addition to validating that the user owns the Token, it can be used for in mechanisms to upgrade software versions or transfer of ownership. In the first instance, license validation by reading the Token at any time interval, say every 600 seconds, user login/logoff, or software start-up. These examples read an existing transaction on the blockchain, but don't create a transaction. For updating information on the blockchain the mechanism is to create a transaction between the user address and the software address that represents the software application. Each transaction that occurs creates a new user address with its own unique public/private key, and a new transaction with data encrypted by the user's new public key. All addresses are unique, with their own public/private key pair. For software license validation purposes, a transaction process from a device with S1 software installed could be like:

1) S1 reads the blockchain transaction for U1
2) S1 decrypts the token from blockchain data for U1
3) S1 checks the Token originates from M1
4) S1 continues to execute on the end user's device

Shown as:

S1.read(UserAddress1.transaction)
S1.decrypt(UserAddress1.Token)
S1.validate(Token)
S1.execute

To upgrade Software application versions such as with a patch update, the software application can periodically request an update from the vendor. The use of a new U2 address for the upgraded software application is so that entitlement to earlier software versions is maintained through U1 in case of downgrade requirements. All Ux addresses are stored in the user's digital wallet, and as such all entitlements are associated with the user. The vendor could also transparently release new license keys with minor releases such as patch updates further reducing any risk of license keys duplication. Software upgrades could be achieved by:

1) S1 sends a request to M1 with new U2 address
2) M1 checks the token came from U1 and is valid
3) M1 creates a new transaction with update data
4) S1 reads data to check if it needs an upgrade
5) S1 auto-upgrades

Shown as:

S1.send(MasterAddress1,UserAddress2,Token)
M1.validate(UserAddress1, Token)
M1.send(UserAddress2, Token)
S1.read(UserAddress2, Token, License, Hash)
S1.upgrade

Although this is similar to existing software version checking mechanisms online such as Microsoft Update, this process allows the software vendor to re-cut a license key or hash code for the software upgrade, and have the software automatically validated. Updating the previous transaction process to include these fields as shown in Figure 4. Both the previous software version and the upgraded software version are available for use.

1) S1 send an update request to M1 with new U2 address

2) M1 checks token came from U1 and is valid
3) MasterAddress2 M2 created for the new transaction
4) M2 cuts new License Key for new software version
5) M2 creates Hash for new software version
6) M2 creates new bootstrap for new software version
7) M2 encrypts the new License Key, Hash and Bootstrap using PublicKey(U2)
8) M2 signs the transaction with PrivateKey(M2)
9) M2 creates new transaction with the new data
10) S1 reads new transaction data for upgrade
11) S1 downloads software and auto-upgrades
12) S1 run itself

Shown as:

```
S1.send(MasterAddress1,UserAddress2, Token)
M1.validate(UserAddress1, Token)
M1.createaddress(M2)
M2.License(License.new)
M2.hash(S1.new)
M2.bootstrap(Bootstrap.new)
M2.encrypt(M2.License)
M2.encrypt(M2.hash)
M2.encrypt(Bootstrap.new)
M2.sign(Transaction.new)
M2.send(UserAddress2, Token)
S1.read(UserAddress2, Token, License, Hash)
S1.upgrade
S1.execute
```

Hence we have shown that license validation can be easily achieved using the blockchain, and through the same mechanism, additional integrity and security protections can be added. Furthermore, blockchain scripts allow for intelligent programming of actions within a transaction. This provides a new level of dynamism as a transaction may take different actions based on the inputs, outputs, field contents and originating and destination addresses. New blockchain protocols are being developed that include full Turing completeness capability, allowing anyone to write smart contracts and decentralised applications with their their own arbitrary rules for ownership, transaction formats and state transition functions (Buterin, 2014).

5.3 Issues that are overcome

We can see that the Bespoke Model overcomes the problems originally highlighted earlier in this paper, and significantly improves on the Master Bitcoin Model. However it does require a separate cryptocurrency ecosystem to be developed and maintained, whilst the Master Bitcoin Model can utilise and run in an existing cryptocurrency ecosystem. While the Master Coin Method meets the requirements outlined for a successful software license validation method and it has been demonstrated to be workable in proof-of-concept, it has limitations that may detract from its usefulness. The Bespoke Model overcomes these as follows:

1) Each software license is hard to copy because the license is represented by a transaction between vendor and user, is cryptographically verifiable, and 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 even have to be disclosed to the user, so it 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 any intermediate parties involved.

3) Software licenses cannot be regenerated because the software application is taking 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) There is no Man-in-the-Middle attack possible using the blockchain. An adversary cannot intercept any data in the blockchain without the user's private key, and cannot redirect DNS or IP traffic to an adversary's custom server to provide software validation.

There are additional benefits beyond the software license validation method, with clear scope for software vendors to provide integrity and protection for their software applications. Furthermore, the blockchain peer-to-peer architecture means that there is no single central point of failure for software license validation. Licensing validators can be run anywhere around the world, and could be run on a not-for-profit basis or on some other commercial model as appropriate. Vendors would run vendor-specific software to manage a license creation process and interaction with the blockchain but will not need to maintain their own dedicated license validation infrastructure with its associated overheads.

The proposed software license validation model provides an opportunity for small developers through to large software vendors to preserve their copyright in their software, and prevent software piracy whilst having a flexible mechanism to license their software.

5.4 Potential Issues

In order for the "Bespoke Model" to work, users will need to provide some form of authentication to their user wallet in order to access the private keys so that the software application can complete its validation function. This is a similar process to users needing to access their blockchain wallet to conduct any transaction in any cryptocurrency, so the action is commonplace. However, if we are to achieve true user mobility where a user can login to any installed application and be validated for use, the wallet will require some portability. In addition, for an automated authentication process to work, the software application will need to access the private keys for the

user addresses to complete the license validation process. Disclosing private keys is not desirable, so the wallet will need to be an application and have the ability to decrypt data on the blockchain and present that to the software application through an API interface. Alternatively, the user may be authenticated to an OpenID or OAuth authorisation service provider such as Facebook or Google, to prove their identity and allow authorised API requests to the user wallet.

Other issues are the security threat model for loss of data or the compromise of the system if a user loses control of a wallet and user credentials are exposed, or a vendor system is compromised. Multi-signature authorisation already provides potential solutions to these issues, similar to two (or more) parties being required to sign a bank cheque. This may place an additional overhead on the software license validation method, but it could also significantly reduce the risk of compromise or loss-of-ownership issues.

5.5 Further Opportunities

In this paper we deal with licensing on the basis of a single user receiving a single license for a software application and where a single user can have many addresses representing software applications in their wallet and on the blockchain. However, in a multi-user corporate environment there are additional challenges, such as licenses that are not permanently allocated to staff and licenses that have to be transferable within the organisation. For example, a staff member leaving the organisation cannot be allowed to exit with a software license in their personal blockchain wallet.

In the Bespoke Model, each license entitlement requires a unique address belonging to the organisation to be sent a Token, with multiple addresses defining the number of Tokens the organisation has available. Hence an organisation with 100 users would have 100 addresses in a wallet dedicated to the organisation. The 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 Active Directory for single sign on. An organisational level blockchain license validation application is required to implement a successful multi-user software license validation method in a multi-user environment.

Additionally, the license validation blockchain method provides an opportunity to manage licenses on non-human operated devices. As the Internet-of-Things grows and evolves, these connected devices will require mechanisms to auto-update software and validate software in a legitimate manner. For example, the customer who owns 10,000 Internet-connected devices, but only pays software maintenance on 2000 of these devices, will only have license keys to update 2000 devices. This capability is easy to achieve in a peer-to-peer decentralised software license validation ecosystem, and hard to manage using any other type of process.

Perkins (2014) states that the Identity of Things is a growing outcome of the Internet of Things. That is, devices and data have a relationship with someone or something that needs to be identified, and assets and users associated with these need to be managed. The license validation method meets the requirements as license entitlement is essentially defined as a Token/source object, providing the identities of the parties through the blockchain address, and the types of activity between parties through the transactional history.

6 Limitations

There are a number of limitations noted in this paper. The blockchain depends on other participants in the cryptocurrency ecosystem to create and validate transactions. However, we do not explore the blockchain ecosystem or business model because there exist myriad types of stakeholders who may perceive various implications or have vested interests in a blockchain ecosystem outcome. These considerations are outside the scope of this paper.

There are currently no standards for cryptocurrency or blockchain technology available, although if the software license validation mechanism was established, standards such as ISO/IEC19770 could be revised to include software validation blockchain technology.

Presently, there is limited peer reviewed work available for cryptocurrency subject matter, and readings are commonly taken from current industry sources and leaders.

7 Conclusion

Software license validation has been one of the primary mechanisms to prevent software piracy since the mid-1970's. The methods for software license validation evolved with the Internet to include online license validation in addition to the traditional paper based license keys provided with software. Software pirates and hackers are able to reverse engineer and remove protection mechanisms whilst license keys are copied, 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 copied or regenerated, 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 software license validation in a cost effective manner through the use of the cryptocurrency theory. The blockchain offers many opportunities to include software integrity and protection mechanisms, providing additional value for software vendors and end users. The blockchain software license validation method can also be automated to provide license validation and identity for Internet of Things devices.

8 References

Business Software Alliance. (2014). The Compliance Gap.
http://goo.gl/9WZYz6. Access 24 August 2014.

Buterin, V. (2014). A Next-Generation Smart Contract and Decentralized Application Platform.

http://goo.gl/enrFst. Accessed 24 Aug 2014.

Fortin, C. (2011). Master Bitcoin - The Proof of Ownership.

http://goo.gl/Tpb0TD. Accessed 24 Aug 2014.

Hochstein, M. (2014). Why bitcoin matters for bankers. American Banker, March 2014.

ISO/IEC. (2012). ISO/IEC 19770-1:2012 Information technology — Software asset management.

Lebo, A. (2014). Implementation of a decentralized, transferable, and open software license system using the Bitcoin protocol.

http://goo.gl/VrFNby. Accessed 24 August 2014.

McCook, H. (2014). An Order-of-Magnitude Estimate of the Relative Sustainability of the Bitcoin Network.

http://goo.gl/M8741r. Accessed 24 Aug 2014.

Morgan, M. J., & Ruskell, D. J. (1987). Software Piracy — The Problems. *Industrial Management & Data Systems, 87*(3/4), 8-12. doi: 10.1108/eb057469

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

http://goo.gl/QTnM0w. Accessed 24 August 2014.

NIST (2001). Introduction to Public Key Technology and the Federal PKI infrastructure.

http://goo.gl/0j5l7N. Accessed 28 October 2014.

Perkins, E. (2014). The Identity of Things for the Internet of Things.

http://goo.gl/v0Ko9l. Accessed 24 August 2014

Suhler, P. A., Bagherzadeh, N., Malek, M., & Iscoe, N. (1986). Software Authorization Systems. *IEEE Software, 3*(5), 34-41. doi: 10.1109/MS.1986.234396