Decentralized Software Licensing Platform(s) with Tokenization & 3rd Party License-exchange capabilities

Ravi Kanth Kaja
Microsoft
e-mail: ravi.kaja@microsoft.com

Abstract

Since the advent of information technology, *software licensing* has become the means for proprietary software publishers to monetize their intellectual property rights. However, reach of the world wide web has made it easier for computer owners to replicate/pirate licenses on multiple machines, providing the intellectual property owners with a tough challenge to build an air-tight process for software licensing and management. This paper presents a *novel – decentralised software licensing system* with a robust mechanism for software licensing, activation, and control of piracy. This framework is envisioned to utilize *tokenized licensing* model, opening doors for a multitude of possibilities, including, but not limiting to, *peer-to-peer, cross-product, cross-enterprise exchange of license tokens* contributing to novel value additions for both software vendors, and consumers alike. Two solution approaches have been explored in this paper that would cater to the needs of a *trust less system* that this framework requires, viz. a *blockchain* based *distributed application*, and *a distributed application* based on *secure multi-party computation protocol.*

Keywords – blockchain, secure multi-party computation protocol, software licensing, tokenization, decentralization

Introduction

Today's software licensing and validation systems in use, do not necessarily ensure that the software publishers are receiving remuneration from all the users of their products/solutions. Once the software is published in the market, it is illegally copied and distributed without permission of its authors. Various research and surveys in the last decade have identified that piracy of information technology products and software has reached as much as 40-50% in most countries. While some robust client-server license validation models have evolved over the years, adapted by large enterprises like Microsoft, Adobe, there is still plenty of opportunities to further tackle piracy, as the pirates evolve to use new means to bypass licenses for their software usage. Further, developing/maintaining/leveraging such a robust software licensing model is not an affordable option for many small-scale vendors.

This paper aims at advocating a new software licensing business model, where any enterprise, small or big, can be onboarded onto a decentralised platform and issue *software license tokens*. Further, the end users, get the flexibility to exchange the license tokens on blockchain for different products, of same or different software vendors. This design can be a robust solution to different forms of software licensing like – perpetual, metering-based-model etc.

Problem Statement(s)

Challenges in preventing piracy and un-warranted use of licenses

A software license activation is usually enforced by various means like – product key activation, subscription based etc. Each of these means have certain attributes that make them vulnerable to unintended/un-licensed users.

- The *perpetual licensing* design often leaves a significant amount of end users (more relevant for individuals than enterprises) with underutilized licensing contracts

- A *product key*, with a characteristic of multiple activation limit can often be shared across web by mal practitioners. There is no control over the medium and reach across which this product key can travel.

- A *subscription*, usually based on an active directory login, is often used by multiple users, by just sharing the credentials across the trusted acquaintances.

Licensing as a solution for 3rd party software creators

Many small and medium scale enterprises cannot afford to develop/maintain a robust software licensing management system and there is still a huge gap in the market for an efficient 3rd party solution that provides this solution while also addressing the 'trust' dependency that the software publishers have on the central entity that provides the services of license generation, distribution, and validation.

The proposed – *software licensing system* addresses the above problem statements in a robust and efficient fashion.

Token-Based Licensing System

In a token-based licensing system, a collection of license-tokens act in place of a single time license activation by a product key (or other alternatives). The license validation module in the software will poll the licensing service at a predetermined rate and will keep burning the license tokens based on the utilization of the software (the usage can be based on any of the properties like active system time, processor utilization, data specific or api calls).

License tokens will be generated when a new software is onboarded on to the platform. Software users can purchase the tokens from the licensing platform based on their usage estimates and integrate the software with their licensing account.

This token-based licensing is essentially an extension of the traditional floating-license mechanism. IBM has implemented token-based licensing for a specific portfolio of their software solutions and have successfully onboarded their customers.

As opposed to the traditional model, which includes a perpetual license, and the newer SaaS (software as a service) model, which is pay-as-you-go, *Tokenized Software Licensing* model offers a third way to license software, one more akin to buying a prepaid cell phone card. The token model allows companies to try something for far less investment than the traditional model.

Value Addition

- In a product key based activation system, a buffer of activation limit is provided to the user to support the scenarios of hardware change, upgrades, and other corner scenarios. A token-based licensing system while providing the same flexibility, also ensures that the previous usage of the software is accounted for by the tokens that are burnt so far.

- Consumers can leverage the flexibility to utilize license token across products, through exchange

- Small enterprises and individual vendors can leverage the software license management platform without having to trust any single central entity

- A customer having a hold of license tokens that can be used across products is more likely to onboard onto other products of the same enterprise as they can leverage the same license tokens – a business advantage to the enterprise

- Onboarding new products, generation of license tokens will be a seamless process through the decentralized platform

To set-up an eco-system that facilitates the exchange of license tokens (that essentially hold monetary value of different exchange rates) belonging to different software from different enterprises requires a robust, trust-less architecture to ensure that no single party can corrupt or misuse the system.

The following sections propose two different approaches to address the above design challenge, leveraging the recent advancements in the fields of block-chain, cryptography.

Design Proposal 1:

[Technology - Blockchain]

Key Concepts & Technology Terms

A quick introduction to a few concepts of *Blockchain* and *proposed Design* are provided in this section.

A *blockchain* is a decentralized series of blocks of transactions that provide special protection against changes using sophisticated cryptographic algorithms.

A *consensus mechanism* is a fault-tolerant mechanism that is used in computer and blockchain systems to achieve the necessary agreement on a single data value or a single state of the network among distributed processes or multi-agent systems, such as with cryptocurrencies.

A *smart contract* is a computer program or a transaction protocol which is intended to automatically execute, control or document legally relevant events and actions according to the terms of a contract or an agreement

A *decentralized application/DApp* is a computer application that runs on a decentralized computing system.

A *blockchain wallet* is a device, physical medium, program, or a service which stores the public and/or private keys for transactions on a blockchain.

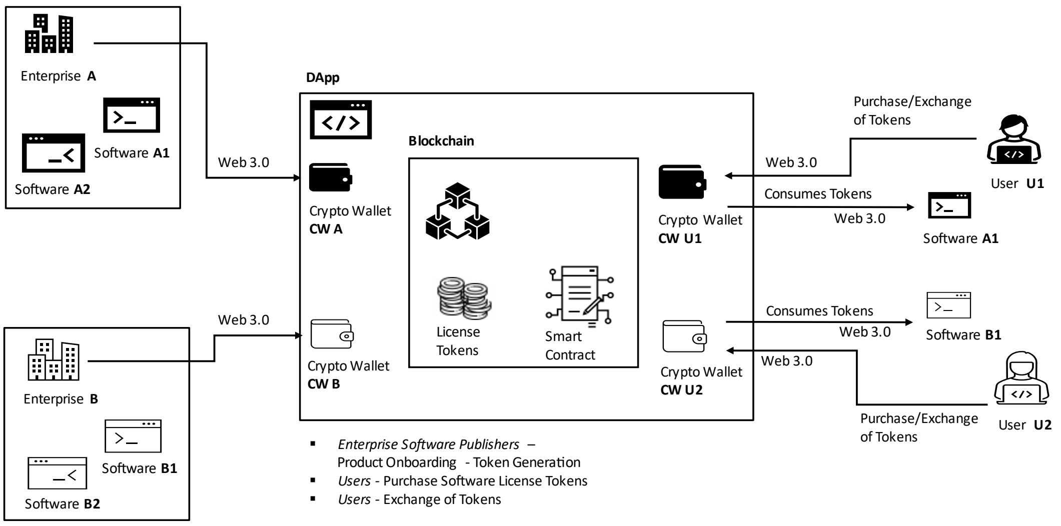
Coin burn is a process where the miners and developers remove the coins from circulation. In other words, coin burn is a process of destroying the coin so that it is not available for further use (trading or otherwise).

Business Model / Solution Design Proposal

A *blockchain based software licensing system* – where tokens are created and distributed through blockchain. A *smart contract* published on blockchain generates tokens every time, an enterprise launches and onboards a new software.

A user can purchase these tokens into user wallet through a DApp. A software that intends to use this licensing system needs to use a *dynamic link library* as part of its installation/activation code. A cryptocurrency wallet address of the user needs to be filled into the software during activation phase. The software keeps polling the wallet for license tokens and burns them as the software is consumed.

User can leverage the DApp to also exchange license tokens of one product with other product. The exchange can span cross-enterprise products as well. This platform acts as a crypto-exchange platform, only, it is used for license tokens.



Tokenized Software Licensing on Blockchain

Entity	**Description**
A	An Enterprise with name 'A'
A1	Software developed and published by Enterprise 'A'
A2	Software developed and published by Enterprise 'A'
B	An Enterprise with name 'B'
B1	Software developed and published by Enterprise 'B'
B2	Software developed and published by Enterprise 'B'
U1	User U1
U2	User U2
CW A	Crypto wallet with address on blockchain representing Enterprise A
CW B	Crypto wallet with address on blockchain representing Enterprise B
CW U1	Crypto wallet with address on blockchain representing User U1
CW U2	Crypto wallet with address on blockchain representing User U2
Blockchain	Blockchain Platform for Software License Management
DApp	A Web App interacting with Distributed App through Web 3.0 framework Will play a role like Software Center
Smart Contract	A smart contract which will execute upon purchase of license token and compensate the software publisher accordingly
License Tokens	*Tokens* that can be consumed by a software as a license validation for a predefined contract

Workflow

A sample workflow is described herewith to elaborate on different scenarios and use cases of this solution:

Software Onboarding and Consumption

1. Enterprise **A** has launched a new software product **A1**
2. **A** onboards on the blockchain-software licensing platform
3. **Smart Contract** is executed and generated **license tokens** for **A1**
4. User **U1** logs-in to the **DApp** which is a software centre and purchases **license tokens** for **A1**
5. The transacted money goes to **A**
6. When user launches **A1,** it asks for **U1**'s wallet address. User enters the wallet address
7. Once **A1** starts running, the **software licensing dll** embedded in it starts polling to the user wallet **CW U1** and starts burning the **license tokens** at a preconfigured rate, based on the smart contract

Exchange of License Tokens

8. **U1** realises that they have more tokens for **A1** left in the wallet, but the user doesn't intend to use **A1** anymore
9. **U1** wants to explore a new software **B2** published by an enterprise/individual software publisher **B**
10. **U1** logs on to the **DApp** and places a request for exchange of tokens with **A1**
11. User **U2** who's in need of tokens for **A1,** but in possession of tokens for **B2** accepts the exchange request from user **U1**
12. **B2** tokens are transferred to the wallet **CW U1**
13. **A1** tokens are transferred to the wallet **CW U2** of user **U2**
14. **U1** will be able to run software **B2** which polls the tokens from **UW U1**
15. **U2** will be able to run software **A1** which polls the tokens from **UW U2**

Design Proposal 2:

[Technology – Secure Multi-Party Computation Protocol]

Key Concepts & Technology Terms

A quick introduction to a few concepts of *Secure Multi-Party Computation (SMPC) Protocol* and *proposed Design* are provided in this section.

Multi-Party Computation [MPC] deals with the problem of jointly computing a function among a set of (possibly mutually distrusting) parties.

A *protocol* is a set of instructions in a distributed computer program.

The basic properties that an MPC protocol aims to ensure are:

Input privacy: The information derived from the execution of the protocol should not allow any inference of the private data held by the parties, bar what is inherent from the output of the function.

Robustness: Any proper subset of adversarial colluding parties willing to share information or deviate from the instructions during the protocol execution should not be able to force honest parties to output an incorrect result.

Basic Roles of MPC:

- The *input parties (IPs)* delivering sensitive data to the confidential computation.
- The *result parties (RPs)* receiving results or partial results from the confidential computation.
- The *computing parties (CPs)* jointly computing the confidential computation.

Each entity involved in an MPC computation may hold one or more of these roles.

Common for all MPC protocols is the concept of '*no single point of trust*', meaning that none of the involved CPs can unilaterally gain access to the encrypted input.

Passive security, which does not guarantee that a corrupt CP computes the intended protocol, but still guarantees privacy of the inputs. With passive security, the CPs that make up the MPC system should be mutually trusted to execute the protocol correctly.

Active security (with abort), which guarantees that a corrupt CP must run the intended protocol, otherwise the protocol will abort with overwhelming probability.

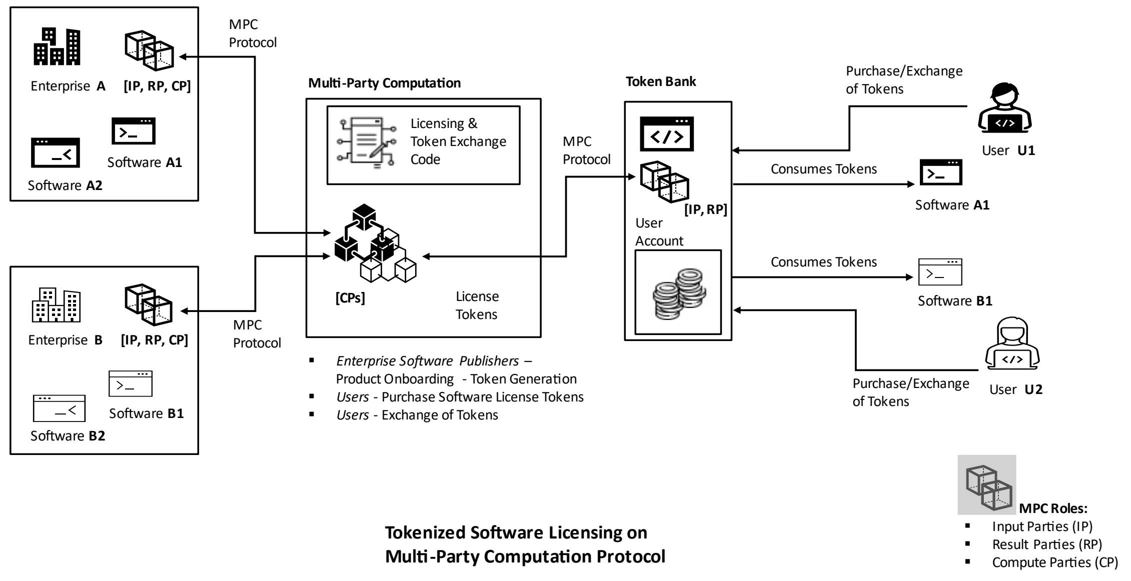
Active security (with fault tolerance) guarantees that the MPC system will continue to operate even if a CP intentionally or unintentionally fails to operate or operate correctly.

Business Model / Solution Design Proposal

A *Distributed Application*, adhering to an *active secure multi-party computation protocol*, where different software vendors, with potentially opposing interests hold different nodes of the system. Activities like *Token Generation, Distribution & Exchange* are facilitated on this platform.

A user is associated with an account (equivalent to a crypto wallet) and can purchase license tokens into this account leveraging the MPC License platform. A software that intends to use this licensing system needs to use a *dynamic link library* as part of its installation/activation code. Software activation is facilitated by configuring/inputting the user's account details from which the software shall consume/burn the license tokens from time to time.

User can leverage this *Distributed Application* platform to also exchange license tokens of one product with other product. The exchange can span cross-enterprise products as well.



Tokenized Software Licensing on Multi-Party Computation Protocol

Entity	Description
A	An Enterprise with name 'A'
A1	Software developed and published by Enterprise 'A'
A2	Software developed and published by Enterprise 'A'
B	An Enterprise with name 'B'
B1	Software developed and published by Enterprise 'B'
B2	Software developed and published by Enterprise 'B'
U1	User U1
U2	User U2
IP	Input Party (Role of MPC)
RP	Result Party (Role of MPC)
CP	Compute Party (Role of MPC)
Multi-Party Computation	A cryptographic protocol based on secret sharing
License Tokens	*Tokens* that can be consumed by a software as a license validation for a predefined contract

Workflow

A sample workflow is described herewith to elaborate on different scenarios and use cases of this solution:

Software Onboarding and Consumption

1. Enterprise **A** has launched a new software product **A1**
2. **A** onboards on the decentralised-software licensing platform
3. An **MPC function** is executed and generates **license tokens** for **A1**
4. User **U1** logs-in to the **Distributed Application** which is a software centre and purchases **license tokens** for **A1**
5. The transacted money goes to **A**
6. When user launches **A1,** it asks for **U1**'s license-account. User enters the account details.
7. Once **A1** starts running, the **software licensing dll** embedded in it starts polling to the user account and starts burning the **license tokens** at a preconfigured rate.

Exchange of License Tokens

8. **U1** realises that they have more tokens for **A1** left in the wallet, but the user doesn't intend to use **A1** anymore
9. **U1** wants to explore a new software **B2** published by an enterprise/individual software publisher **B**
10. **U1** logs on to the **Distributed Application** and places a request for exchange of tokens with **A1**
11. User **U2** who's in need of tokens for **A1,** but in possession of tokens for **B2** accepts the exchange request from user **U1**
12. **B2** tokens are transferred to the license account of **U1**
13. **A1** tokens are transferred to the account of user **U2**
14. **U1** will be able to run software **B2** which polls the tokens from the account of **U1**
15. **U2** will be able to run software **A1** which polls the tokens from account of **U2**

Blockchain Solution vs SMPC Solution

While the approach varies in the 2 proposed solutions, both the solutions provide a robust trust-less environment that is required for the collaboration of different parties of conflicting/opposing interests.

Both, Blockchain, and Multi-Party Computation are built on the building blocks of cryptographic concepts.

While a *private blockchain*, where each of the participating enterprises share nodes in the network can be one possible solution, leveraging a *public blockchain* will make the platform open to individual, small & medium scale content/software creators without the pre-requisite for them to participate in the blockchain network for trust-guarantee. A *private blockchain* could mean no additional *gas fee* for the transactions, but also an overhead of maintaining the blockchain ecosystem. New blockchain networks like MATIC which run on *EVMs (Ethereum Virtual Machines)* provide an alternative for the more commonly used networks like *Ethereum.* They provide low *gas fee*, faster *transactions times*. Another predicament of a blockchain based solution is the number of computations, carbon foot-print concerns, and the lack of control over the network in a rather semi/un-stable world of blockchain, where the acceptance and adoption are still in its initial stages.

A Secure *Multi-Party Computation (SMPC) Protocol* based solution, doesn't share the concerns of the alarming carbon-footprint that it's alternate solution shares. However, this approach is expected to significantly involve a lot more effort in implementation, to analyse, and leverage the recent advancements in this field, to come up with an efficient solution with practically acceptable response times. In the *public blockchain* based solution, software users maintain *crypto wallets*, which are tightly integrated with the blockchain network, making them equal stakeholders in the trust-framework provided by the system. Whereas, in the *SMPC* based solution, user accounts are part of the distributed application, that is run by a network of *Compute Parties*, potentially owned by different software publishers, not necessarily sharing the same interests as that of end users, this solution is more like a *private blockchain* based solution.

Market Relevance and Opportunity

Various players in the market are actively exploring different strategies to utilize tokenization & decentralized systems for software licensing.

- EY & Microsoft have collaborated to work on a blockchain based solution for content rights and royalty management of media in entertainment industry
- Jengascoin has launched an NFT based software licensing solution for digital assets
- Multiven-Open Market place is using a blockchain based solution to validate license ownerships
- IBM has adapted token-based software licensing for some of their software suites and has been successful in optimizing the licensing for its customers

More Opportunities

While licensing is one aspect of prevention for software piracy, ***a more regulated distribution of software*** is another means through which distribution & usage of pirated copies of a software can be prevented. There are multiple strategies being explored in this aspect as well, where a blockchain network is leveraged to ensure controlled distribution/initiation of software.

References

[1] Buterin, V. (2014). A Next-Generation Smart Contract and Decentralized Application Platform. http://goo.gl/enrFst

[2] Master Bitcoin - The Proof of Ownership. http://goo.gl/Tpb0TD

[3] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. http://goo.gl/QTnM0w

[4] J. Herbert and A. Litchfield, “A Novel Method for Decentralised Peer - to - Peer Software License Validation Using Cryptocurrency Blockchain Technology,” Proc. 38th Australasian Computer Science Conference

[5] Aaron Lebo, Implementation of a decentralized, transferable, and open software license system using the Bitcoin protocol | GitHub

[6] From Keys to Databases—Real-World Applications of Secure Multi-Party Computation David W. Archer, Dan Bogdanov, Yehuda Lindell, Liina Kamm, Kurt Neilsen, Jakob Illeborg, Pagter, Nigel P. Smart, And Rebecca N. Wright

[7] The Simplest Protocol for Oblivious Transfer Tung Chou and Claudio Orlandi Technische Universieit Eindhoven and Aarhus University

[8] Multi-Party Computation with Omnipresent Adversary - Hossein Ghodosi and Josef Pieprzyk

[9] Rationality and Adversarial Behavior in Multi-Party Computation (Extended Abstract) Anna Lysyanskaya and Nikos Triandopoulos