Tokenized Micro-License

Technical Brief

Version: 1.1
Author: Joe Roets
Last Update: 3/19/2018

Software License Model History

To understand the Dragonchain Tokenized Micro-License, one needs to understand and consider the history of software licensing.

In the early software industry, most software produced was shared often as public domain software. With the advent of proprietary software in the late 1970s and 1980s, software was sold under license for use. These licenses were refined in the industry to suit the availability of computing power.

Version Licensing

A typical model was to sell license to use a particular version of a software product. With the connectivity of the Internet, software could soon be updated readily by the vendor, many software products were licensed for all updates for a major version of the product. That is, the user would have license rights to obtain and use all updates to a product until a new, major version was released. After a new release, they could continue to use the old and unsupported version or purchase an upgrade. This offered added flexibility to the consumer and vendor. The vendor could make much needed updates to fix bugs found after release and provide non-major upgrades. The consumer would get some level of new value in minor version of the product and also physically held the software to use at any time in the future. Some examples of the use of the version based licensing model are Microsoft Windows 3.1 and Adobe Photoshop 7.0.

Subscription Licensing

In the 1990s and 2000s, many software vendors began to use the approaching ubiquity of the Internet to provide subscription based licensing, wherein the consumer pays on a time basis for access to software. In this model, the user will typically not hold the software or run it on a local

machine. Instead, the software is managed and executed on centrally managed hardware by the vendor. This allows for very simple updates to the software in place in a consistent manner and will generally lower maintenance cost to the consumer and vendor. The vendor can control access to the services as well, and provide the ability to provide feature driven pricing models for the services and particular advanced features. The negative for the consumer is generally in the loss of right and possession of the software. The model generally represents the shift from licensing rights to *own* the purchased software and *possess* it physically to a *utility* focused right. That is, the consumer has a right to access the service or its underlying application software interfaces (API). The consumer never possesses the software or its executable code. If the consumer stops paying for the subscription, access is withdrawn and is unavailable.

Among other issues, this model did not standardize or solve the resulting issue with ownership of the consumers' data on the system. That is, when using a subscription license, how does a user retain control of the data produced or held in the subscription based license system? If the system stores the data in the vendor's infrastructure (very typical due to efficiency), then the consumer will need some ability to extract their data from the vendor's storage. Some examples of the use of the subscription licensing model are Google GSuite, Microsoft Office 365, and Adobe Creative Cloud.

The Tokenized Micro-License™

Dragonchain's Tokenized Micro-License (TML) was created to provide a new model for software access. It allows the local holding of licenses much like the early models, yet allows for vendor or decentralized hosting of the software services. It also standardizes for many flexible forms of redemption via software access and execution. For the vendor, licensing is more flexible, and powerful anti-piracy measures are possible. Updates to services can be controlled much like in the subscription model. For the consumer, a user is not paying for software utility that isn't being used. Ceaseless passing of time does not itself penalize.

License ownership is recorded on the blockchain and decentralizes ownership and control. Asymmetric encryption technology (public key cryptography) is used to enable the consumer to physically hold the key which "own" the tokens.

TML Service Interaction

The token itself contains a license that is maintained on the blockchain with both programmatic (smart contract based) terms and human readable (traditional legal) terms embedded. The token's license interacts on every use, execution, or access with a license embedded into every service (programmatic/smart contract and traditional) to create a very flexible framework for software license innovation.

The terms for both the token and all services can be updated on the blockchain in a manner that allows anyone to prove and/or verify that a particular execution occurred under a particular set of license terms (or other legal arrangement). This model also allows for a very flexible update

arrangement wherein a vendor may do things such as specify license update frequency, specify terms for notice on updates, or allow consumers to vote on terms or feature updates. This model can also solve issues with access to data stored on the vendor's system as the public keys of the owner may be allowed access to their owned data even if they no longer hold license access to the system services.

This can more directly model the commoditization of service utility in the "Software as a Service" or "Platform as a Service" hosting models so prevalent today. It allows for commodity service micro-payments (e.g. Amazon AWS, Google Cloud, MS Azure) to be modeled into the license itself and to be controlled in a more granular fashion. It also allows the ability to track and audit billing in a provable and transparent manner.

Patent

The patent was granted in November 2018 (US Patent #10,135,607).

