1208 Wharf Street
Victoria, British Columbia
V8W 1T9
Duncan Hogg
Co-op Coordinator
Faculty of Engineering
University of Victoria
P.O. Box 1700
Victoria, B.C.
V8W 2Y2

Dear Duncan,

Please accept the accompanying Work Term Report entitled "Creating a proof of concept decentralized marketplace on Ethereum."

This report is the result of work completed at Membran Entertainment Canada. Leveraging my existing knowledge of blockchain concepts with my interest in programming, I learned how to build decentralized applications using frameworks such as truffle, deploy to testnets such as ropsten and use reactive stores such as Drizzle to manage contract state.

This report is the result of interest and exposure to blockchain development and technologies as part of my third co-op work term. Using the most popular smart contract programming language, Solidity, I created logic for a decentralized application. Some issues of implementing and determinating the viability of smart contracts to simplify transactions include regulation, conflict resolvement and ability to reverse fraduent transactions. Technological advances including exponential increases in computational power that render modern cryptographical useless are not explored. Some problems faced in this report include rapid changes in technology, constantly changing standards, and legal uncertainty surrounding blockchain.

I would like to thank my manager, Dino Coletti, for his patience and good judgement, as well as the my coworkers who were always willing to help.
Sincerely,

David Li

Table of Contents

List of Figures

List of Tables

Summary

The rise of centralized music streaming platforms has significantly reduced artist compensation because incentives to pay music labels first, unclear payment structures, and a lack of transparency. Another way to compensate artists is to issue digital tokens, which are spendable on accommodations, recording time and studio time. In order to empower artists and avoid pitfalls of streaming services, ownership and control of digital token must be decentralized.

Although **blockchain** is the technology that is associated with the cryptocurrency, Bitcoin, the use cases of blockchain are numerous including building decentralized applications. Ensuring that digital assets can be redeemed requires the usage of **smart contracts** to automatically execute transactions. Additionally, ensuring ownership of tokens and safe transfers of digital assets are essential.

Visually representation of rewards can be implemented using an image, hosted of decentralized storage such as IPFS, or contain a form of "dna" that maps to specific components.

Glossary

blockchain A blockchain is a digitized, decentralized, public ledger of all cryptocurrency transactions. It constantly grows as the most recent transactions (blocks) are appended in chronlogical order. iv, 1, 2, 5, 10, 12

Dapp Decentralized Application have backend code running on a decentralized peer-to-peer network and not controlled by a single entity. In a decentralized application transactions are versified through consensus of multiple users. 1

Ethereum an open software platform based on blockchain technology that enables developers to build and deploy decentralized applications. The main unit of currency is ether. The most popular block explorer, etherscan is useful for analyzing transactions and viewing general Ethereum statistics. 1–3, 7

gas In Solidity, your users have to pay every time they execute a function on your DApp using a currency called gas. Users buy gas with Ether (the currency on Ethereum), so your users have to spend ETH in order to execute functions on your DApp.

How much gas is required to execute a function depends on how complex that function's logic is. Each individual operation has a gas cost based roughly on how much computing resources will be required to perform that operation (e.g. writing to storage is much more expensive than adding two integers). The total gas cost of your function is the sum of the gas costs of all its individual operations. 7

HyperLedger group of open-source blockchain technologies started by the Linux Foundation 1

smart contract computer program that directly controls transfer of digital currencies or assets under predefined conditions and used to automatic transactions on the blockchain. These transactions are trackable and irreservable. iv, 1, 2, 7

1. Introduction

Table 1 Timeline of Cryptocurrency

2008	Bitcoin White Paper
2009	Bitcoin Genesis Block
2013	1 BTC = $ 31 USD
2013	**Ethereum** White Paper
2015	**Ethereum** Genesis Block
2015	**HyperLedger** starts
2017	Over 1000 different cryptocurrencies
2018	AWS Blockchain Templates

In 2008 bitcoin white paper [1] described a way to solve the double spending problem without a centralized body using **blockchain**. Although, the value of bitcoin (BTC) has grown exponentially, high computational and energy consumption in mining and slow performance [2]. Released in July 30, 2015, Ethereum, an open-source platform based on blockchain technology, distinguishes itself from bitcoin through faster transactions, unlimited processing capability for **smart contracts**, and its network is optimized to support **Decentralized Application(DApp)** [3].

1.1. Harvest Rewards Program

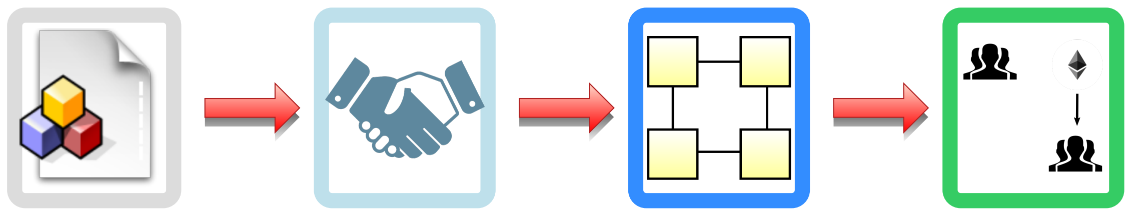
The prevalence of music streaming platforms has been great for its ability to expose listeners to new music in an easy and accessible way, but the shift from physical music sales to digital streaming has created tension and confusion in the music industry. Artists may receive compensation based on total listens from streaming services, but there are many stories which outline how little profit artists actually receive from these royalties. This is an issue even for massively successful artists, let alone emerging or independent artists. As traditional record sales decrease, artists will become increasingly reliant on revenue from streaming platforms. The current model cannot meet this demand and so a new method of compensation and exchange must be developed to allow the music industry and economy to have continued growth while not leaving behind the artists and creators that drive the industry.

Goal of System The Membran Entertainment Canada Rewards Program harnesses the Ethereum blockchain to provide a tool that music businesses can use as an additional incentive for their artists and clients. The goal is not to fix the royalty system, but to instead add a new method by which artists can be compensated for plays of their music. In theory, this program can create a new exchange economy where musicians and artists can reinvest the value of streaming plays into their product and brand.

Any participating music businesses could implement this program. They will be able to issue digital tokens to their roster of artists using their own measurements, but the recommended method would be 1 Token for 1 Play. This can be done in addition to or instead of traditional streaming royalty payments, at the discretion of the music business. Tokens will be distributed on a regular basis (monthly, quarterly, etc.) to the participating artists. Digital assets such as Ether, and tokens are stored in an Ethereum wallet such as metamask.

1.2. Smart Contracts

Traditional legal contracts are written to represent the contracting parties. In a smart contract, self-executing source code is used to automatic transactions that are publicly available on the blockchain [3]. Furthermore, **smart contracts** allow buyers and sellers exchange money, property, shares, or anything of value in a transparent, conflict-free way while avoiding the services of a middleman. This allows validation of complex transactions swiftly while maintaining transparency. Although the benefits of using smart contracts are obvious, legal enforceability is difficult because "no central administering authority to decide a dispute" exists [**keyfindings:Online**].



SOURCE CODE	TRANSPARENT	BLOCKCHAIN	PAYMENT
self-executing code that faciliates agreement between parties	terms of contract are available on the public ledger	event triggers the execution of the contract (redeeming a ticket)	accounts are updated, parties recieve amounts based on contracts

Figure 1: Illustrating how a smart contract works

In addition, irreversible and immutable transactions are a disadvantage that hackers can exploit. For example, an amateur coder killed the contract that allowed users to transfer Ether for the Parity **Ethereum** Wallet, rendering 150 to 300 million dollars completely useless [4]. Scrutinizing smart contracts and reducing bugs in production code is essential. The most common token standards, ERC20 and ERC721 interfaces outline how tokens work on Ethereum (See Appendix A).

The desirable properties of a **blockchain** such as integrity and immutability of data, e.g., transactions and smart contracts, require an active mining pool. In a proof of work (POW) consensus mechanism, miners need to invest energy and resources, i.e. computational power, to participate in the consensus process and prevent tampering blocks. Additionally, miners are incentivized to run nodes that verify transactions.

1.3. Representation of Digital Assets on the Blockchain

ERC20 Token Standard The ERC20 token standard allows any tokens on Ethereum to be re-used by other applications: from wallets to decentralized exchanges. This is the most commonly used token on the Ethereum network at the moment.

ERC721 Token Standard Non-fungible tokens are often used to represent scarality on the **Ethereum** network. The following standard (See Appendix A) allows for the implementation of a standard API for NFTs within smart contracts. This standard provides basic functionality to track and transfer NFTs. NFTs can represent ownership over digital or physical assets. As seen in cryptozombies, this can also contain "dna".

Other Smart Contracts Other contracts in the Harvest system, such as the storefront contract allow creation of reward tokens (ERC721s), however, since deploying ERC721s is gas intensive, a separate reward deployer contract is instantiated and transfers ownership to the storefront. As shown in Figure 5 better distributes the gas cost of deploying contracts. Additionally, the reward and storefront contract contain the ownable property (owned by the contract creator).

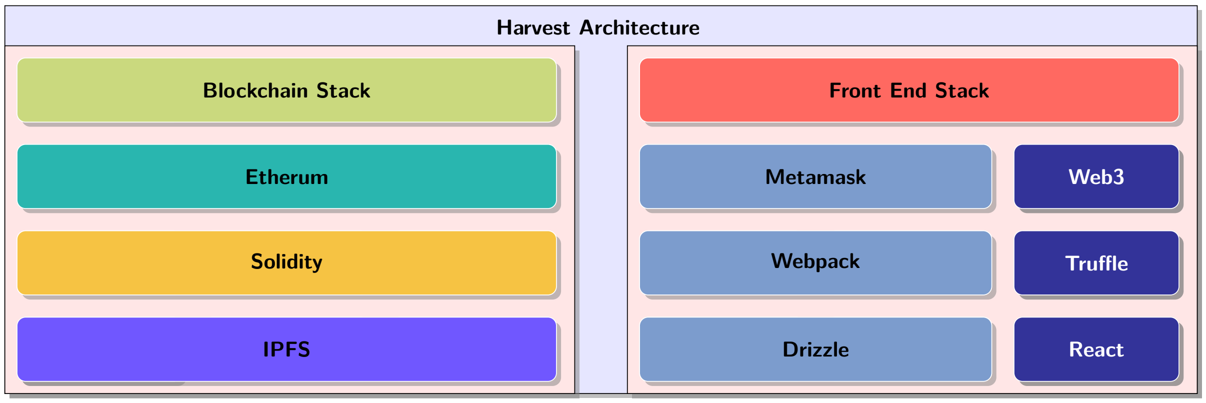


Figure 2: Architecture for smart contracts and front-end for harvest application

Although determining the digital value of assets is challenging, the ability for a users to trade assets between themselves is essential for a decentralized marketplace. Implementation of a exchanger contract to buy/sell rewards and currencies are outside the scope of a proof of concept application.

2. Discussion

The tools used in the harvest application are, a firefox/chrome plugin, metamask for connecting to the ethereum blockchain and testnets, the most popular framework for smart contract development truffle, and drizzle, which manages contract state.

2.1. Decentralized Applications

Decentralized systems are inherently more reliable, currently face scalability issues, and are trustless systems [1, 3]. Fulfillment of smart contracts is limited due to off-chain information, logistical challenges in obtaining impartial or truthful inputs from stakeholders, and verification of transaction completion involving real-world assets. This suggests effectiveness of smart contracts is currently limited to digital assets rather than for physical assets. In addition, translating legal-binding contracts to code is challenging because the majority of programmers lack a legal background.

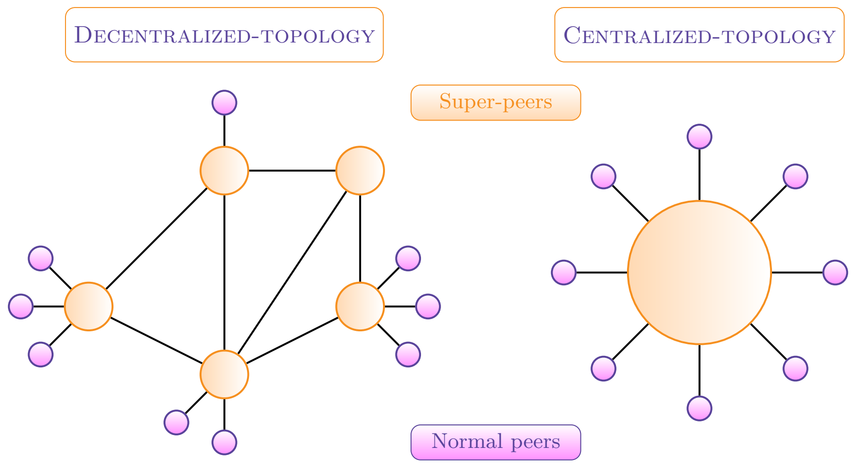


Figure 3: Decentralized structure vs Centralized Structure

Despite shortcomings that involve transactional steps that are computational impossible to verify (package is delivered), performance of smart contracts is superior as transactions are tamper-proof. Additionally, reduction of ambiguity from usage of smart contracts is highly probable because only one interpretation is possible. Coding errors or bugs exist in every non-trivial application, therefore transactions through smart contracts intrinsically carry some risk. This implies protocols and prior real-world agreements are necessary to migrate risk when executing smart contracts. Furthermore, increased precision and detail for transactional inputs and outputs required for creating smart contracts would benefit all parties.

2.2. Trading digital currency for rewards

Artists will receive their tokens to a wallet where they can store and save or spend their tokens. Artists will be able to browse through the music business catalogue of goods and purchase items or services using their tokens.

What separates this project from a traditional rewards program (for example Airmiles or Aeroplan) would be the ability for artists (or any other token holder) to exchange tokens between each other easily using the Ethereum blockchain. Much of the work taking place in the music industry is conducted in exchange or in kind. The MEC Rewards Program could allow for artists to pass a value back and forth in exchange for tokens. The emergence of blockchain technology allows for these kinds of interactions to be transparent, trackable and scalable.

Algorithm 1 Trading ERC20 tokens for ERC721 Rewards

```
    Input uint256 _ERC20Num, HarvestToken ERC20, RewardToken ERC721
    Output Number of ERC721 rewards returned
Require:
    More than one ERC721 must be traded.
    Transaction sender has _ERC20Num number of ERC20 tokens.
    Make sure enough ERC721s are available (not owned by others).
    Permission to spend _ERC20Num number of ERC20 tokens.

1: rate← $ERC721.getRate()$                      ▹ gets rate of ERC20 to ERC721
2: $numOf721s$ ← _ERC20Num / rate
3: ERC20.transferFrom(msg.sender,owner,_ERC20Num)       ▹ owner is global variable
4: $i$ ← distributed_token721    ▹ Number of distributed 721 tokens is a global variable.
5: repeat
6:     ERC721.transferFrom(owner,msg.sender,i)
7: until $i > i + numOf721s$
8: distributed_token721 = i
       return $numOf721s$
```

Receiving rewards in Harvest requires an artist to approve the StoreFront contract to spend their ERC20 tokens with the expectation that a digital representation of a reward (ERC721) will be received. This requires the specific reward which could be recording time, or studio time to exist. In addition, only the owner of StoreFront can create new rewards and different kinds of rewards. Furthermore, in order for redemption to occur, real-world verification is necessary, then the result are recorded on the **blockchain**. Since transactions cannot reversed adding thoughtful assert statements (reverse changes caused by ongoing transaction) is essential.

Assuming that ERC721 tokens will never be burned (destroyed), creating new tokens is simple. Since every ERC721 token has a unique id in its contract, incrementing the token id based on total supply of ERC721 is efficient.

Algorithm 2 Reward Creation Algorithm

```
    Input RewardToken ERC721, uint256 numTokens
    Output Returns true if new tokens are minted.
Require:
    Make sure the 721 reward token is registered (not counterfeit).
    Only the owner of a storefront can create rewards.
1: $newTokenId = ERC721.totalSupply()$
2: repeat
3:     ERC721.mint(address(this),i)        ▷ Reward are owned by storefront.
4: until $i = newTokenId > i + numTokens$
5: distributed_token721 = i
        return True
```

2.3. Rendering Rewards

Mapping unique info to components

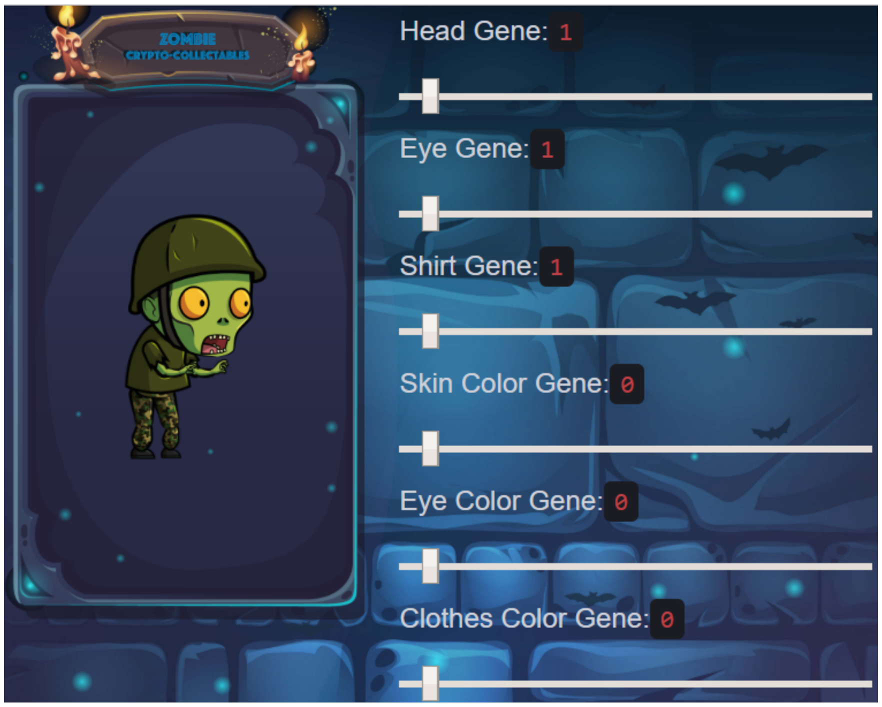


Figure 4: Adjusting the appearance of the zombie by changing DNA values

Although ERC20 tokens are used as a to represent currency, representation of rewards can be done using ERC721. Since each ERC721 token has a unique token identifier, they can be displayed as digital collectables such as etheremon and cryptozombies [5] with their "dna" mapping to specific components.

As described in the popular solidity tutorial cryptozombies [5], a zombie's appearance will be based on its "Zombie DNA". Zombie DNA is simple it's a 16-digit integer, like:

$$\underbrace{83}_{\text{head}} \quad \underbrace{56}_{\text{eyes}} \quad \underbrace{281049284737}_{\text{Other parts}}$$

Just like real DNA, different parts of this number will map to different traits. The first 2 digits map to the zombie's head type, the second 2 digits to the zombie's eyes, etc. One approach for rendering unique tokens in harvest, is to specific the type of reward which maps to an icon, and the colour values for the icon.

$$\underbrace{83}_{\text{type}} \quad \underbrace{99}_{\text{red}} \quad \underbrace{23}_{\text{blue}} \quad \underbrace{13}_{\text{green}}$$

Despite simplicity of using predefined icons, distinguishing between different rewards is not guaranteed.

Using Decentralized Storage (IPFS)

Allowing storefront owners to upload their own images that represents their reward is desirable. Although decentralized storage such as IPFS or Swarm is not required for image and text storage as the smart contract at minimum, could contain a url, but having immutable, permanent links is beneficial. Large amounts of data can be addressed with IPFS, and the immutable, permanent IPFS links can be placed into a blockchain transaction. This timestamps and secures the content, without having to put the data on the chain itself.

Information to access the image (immutable and permanent IPFS links) is included inside a blockchain transaction. This secures the content without having to store the data on the chain which is quite expensive.

2.4. Optimization of smart contracts

On the **Ethereum** network uses pay for transactions using **gas**, this highly encourages programmers to create efficient **smart contracts** that minimize costs to users while adhering to restrictive limitations for developing smart contracts. Performing calculations off-chain and verification in **smart contracts** is highly desirable because writing to the blockchain is expensive. Balancing security, efficiency and simplicity is challenging, but extremely important as transactions cannot be reverted, and source code is publicly available. Currently, on Ethereum transactions sizes are limited to 32 kb, and **smart contracts** cannot be larger than 24 kb.

Cost of Common Operations

Operation	Gas	Description
ADD/SUB	3	Arithmetic operation
MUL/DIV	5	Arithmetic operation
ADDMOD/MULMOD	8	Arithmetic operation
AND/OR/XOR	3	Bitwise logic operation
LT/GT/SLT/SGT/EQ	3	Comparison operation
POP	2	Stack operation
PUSH/DUP/SWAP	3	Stack operation
MLOAD/MSTORE	3	Memory operation
JUMP	8	Unconditional jump
JUMPI	10	Conditional jump
SLOAD	200	Storage operation
SSTORE	5,000/20,000	Storage operation
BALANCE	400	Get balance of an account
CREATE	32,000	Create a new account using CREATE
CALL	25,000	Create a new account using CALL

Table 2: Cost of common operations in the Ethereum Virtual Machine

In addition, the gas limit for deploying smart contracts varies based on consensus with the miners, for example, on July 27, 2018, the gas limit for ropsten was 4.7 million units, but two days later on July 29, 2018 it reached 9.4 million units. Raising the gas limit is possible through increasing the amount of computation power available in the network, this may result because developers want to test large smart contracts.

Oftentimes, in smart contract development, extending and customizing an existing contract (ERC20 and ERC721) is beneficial. Although some smart contracts may be deceitful small, inheritance and factory contract creation functionality can easily reach the gas limits set by the miners.

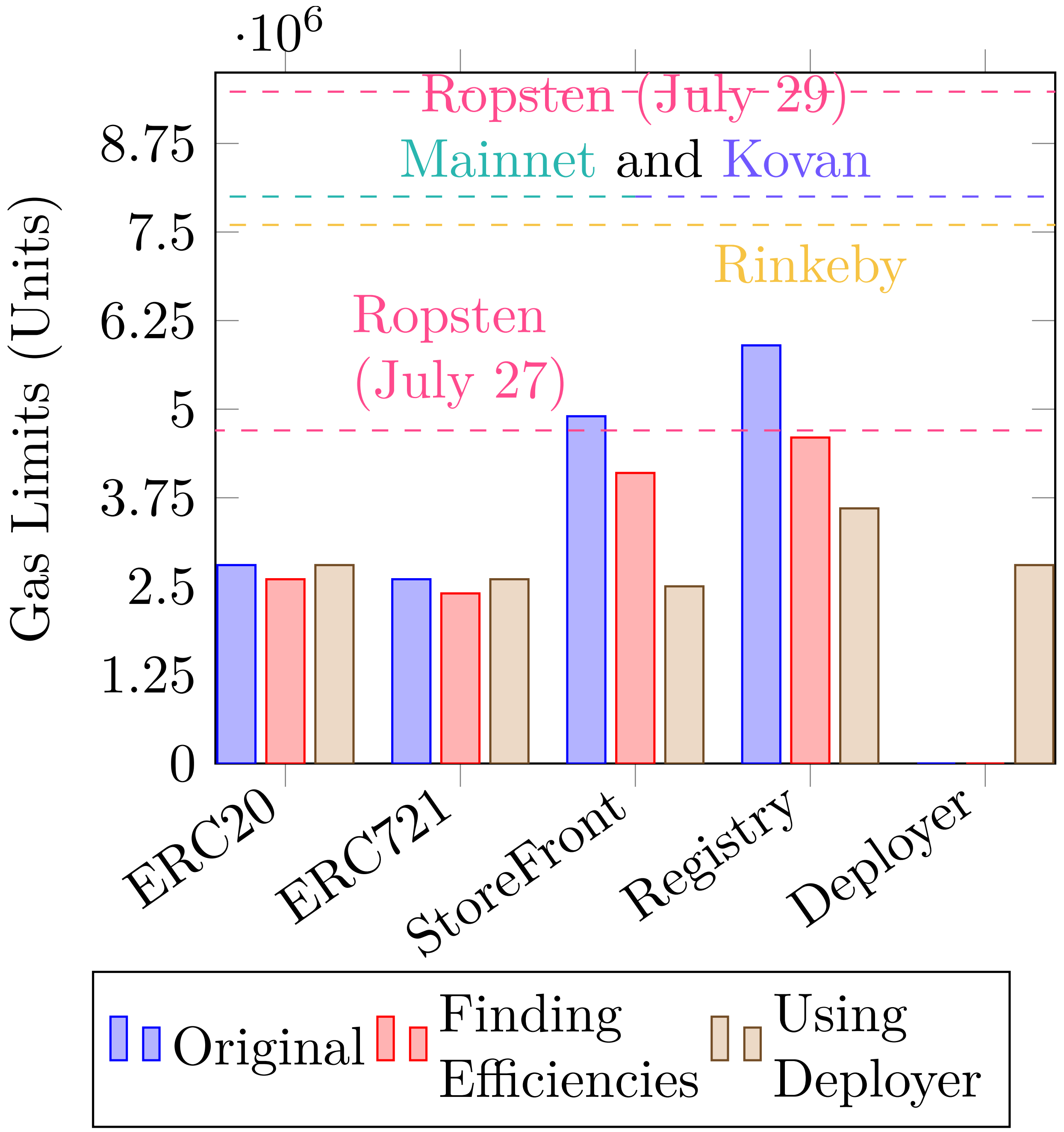


Figure 5: Optimization of Harvest Contracts and Gas Limits for Ethereum networks

As shown in Figure 5 the original smart contracts were inefficient, reducing expensive operations (see Table 2) such as storage and contract creation for factory contracts (deploying ERC721s) allows deployment on a testnet with for a relative low limit of 4.7 million units.

Previously, the Registry contract allowed creation and tracking of storefront contracts which in turn could create rewards, this resulted in a very costly Registry contract 5. In

Solidity, creating smaller contracts with specific functionality is highly encouraged, for example having a reward deployer contract that only produces a new ERC721 contract and transfers ownership is highly efficient.

2.5. Limitations of Smart Contracts

Continued research into **blockchain** technologies is necessary as innovations mitigate scalability issues, widespread adoption of cryptocurrency and technologies evolutions address the shortcomings of blockchain. In trustless systems tampering or modification of transactional history is extremely difficult. This suggests that smart contracts can greatly improve transactions, however, issues including scalability, reversing fraudulent activity and reduction of contract deployment hinder mass adoption.

All computer programs have bugs, by limiting complexity of smart contracts, using reliable standards highly audited packages, and using software security analyze tools risk can be greatly reduced.

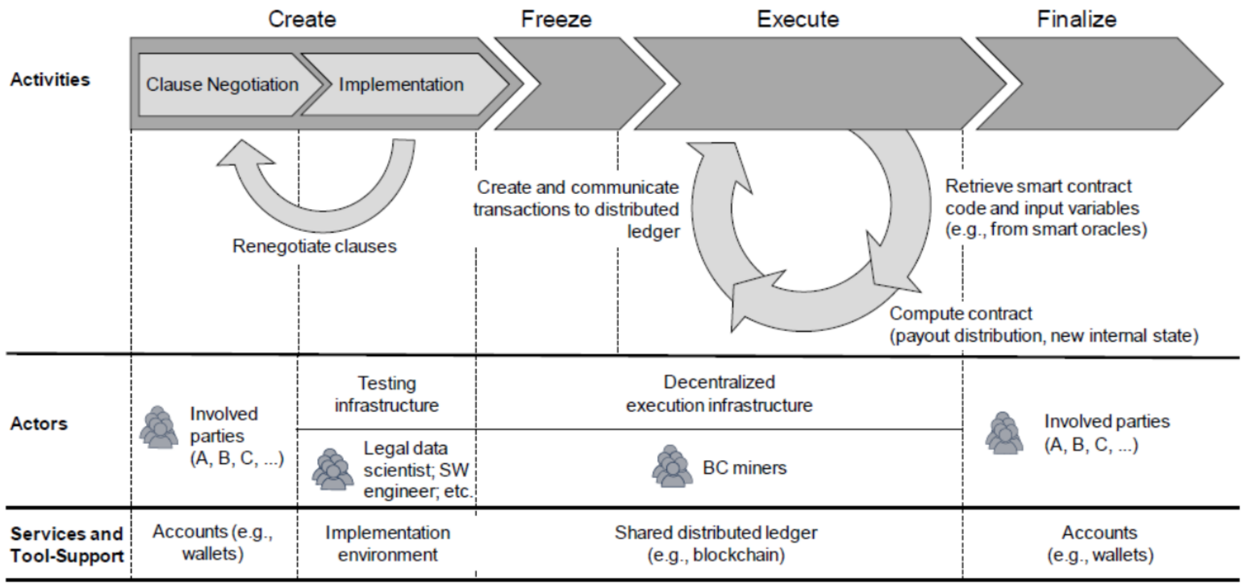

Create
Freeze
Execute
Finalize
Activities
Clause Negotiation
Implementation
Renegotiate clauses
Create and communicate transactions to distributed ledger
Retrieve smart contract code and input variables (e.g., from smart oracles)
Compute contract (payout distribution, new internal state)
Actors
Involved parties (A, B, C, ...)
Testing infrastructure
Legal data scientist; SW engineer; etc.
Decentralized execution infrastructure
BC miners
Involved parties (A, B, C, ...)
Services and Tool-Support
Accounts (e.g., wallets)
Implementation environment
Shared distributed ledger (e.g., blockchain)
Accounts (e.g., wallets)

Figure 6: Lifecycle of Smart Contract Creation [6]

As shown in figure 6 multiple parties are required to create comprehensive smart contract, input from legal professions and negotiations are required. Additionally, unlikely other forms of software, iterative releases of **smart contracts** is inefficient.

3. Conclusion

Blockchain technology is disrupting many industries. The economy of the music industry has yet to recover from the last disruption, the proliferation of digital music first through mp3s and now streaming platforms. The Ethereum network provides key standards such as ERC20 and ERC721 to represent digital currency and scarcity in the blockchain. Understanding ownership and nuances of Solidity is key to creation of bug-free and efficient smart contracts. As shown in Figure 5 shifting specific functionality such as factory contracts is essential to meeting the gas limits.

Artists may receive compensation based on total listens from streaming services, but there are many stories which outline how little profit artists actually receive from these royalties. This is an issue even for massively successful artists, let alone emerging or independent artists. The MEC Rewards Program aims to use new technology (the blockchain and decentralized apps) to bring transparency and innovation to the compensation of artists in the new digital economy, while creating a valuable business proposition for industry. Representation of digital rewards could involve storing an image online and including that url inside of a smart contract. Overall, blockchain is an effective platform for a decentralized marketplace.

4. Recommendations

Additionally, storage and contract creation is expensive, so optimization of smart contracts is essential. Furthermore, usage of **smart contracts** completely automated transactions are limited to computational verifiable events, otherwise users could falsify input data. As adoption of blockchain technology grows, and scalability of the Ethereum network increases, decentralized applications will become a viable alternative.

Artists will receive their tokens to a wallet where they can store and save or spend their tokens. Artists will be able to browse through the music business catalogue of goods and 'purchase' items or services using their tokens. By adding an additional compensation and value to the way they compensate artists for plays, they will be able to create more loyalty and become even more attractive to potential artists.

Provided a deterministic set of inputs and outputs, with computational verifiable conditions, using smart contracts is beneficial. Increasing awareness about **blockchain** technologies, so that the average person understands the limitations of smart contracts and suitable applications is a key aspect in increasing adoption and widespread acceptance of **blockchain** technologies.

5. References

[1] *Bitcoin White Paper*. [Online] Available: https://bitcoin.org/bitcoin.pdf. Accessed April 25, 2018.

[2] Saeed Elnaj. *The Problems With Bitcoin And The Future Of Blockchain*. [Online] Available: https://www.forbes.com/sites/forbestechcouncil/2018/03/29/the-problems-with-bitcoin-and-the-future-of-blockchain. Accessed May 06, 2018.

[3] *Ethereum White Paper*. [Online] Available: https://github.com/ethereum/wiki/wiki/White-Paper. Accessed April 25, 2018.

[4] Thijs Maas. *Yes, this kid really just deleted 300 MILLION by messing around with Ethereums smart contracts*. [Online] Available: https://hackernoon.com/yes-this-kid-really-just-deleted-150-million-dollar-by-messing-around-with-ethereums-smart-2d6bb6750bb9. Accessed May 29, 2018.

[5] Loom Network. *Cryptozombies Solidity Tutorial*. URL: https://cryptozombies.io/.

[6] Christian Sillaber and Bernhard Waltl. "Life Cycle of Smart Contracts in Blockchain Ecosystems". In: *Datenschutz und Datensicherheit - DuD* 41.8 (2017), pp. 497–500. ISSN: 1862-2607. DOI: 10.1007/s11623-017-0819-7. URL: https://doi.org/10.1007/s11623-017-0819-7.

A. Token Standards

Listing 1: ERC20 Token Standard Interface

```
pragma solidity ^0.4.20;
contract ERC20Interface {
    function totalSupply() public constant returns (uint);
    function balanceOf(address tokenOwner) public constant returns (uint
        balance);
    function allowance(address tokenOwner, address spender) public constant
        returns (uint remaining);
    function transfer(address to, uint tokens) public returns (bool success);
    function approve(address spender, uint tokens) public returns (bool
        success);
    function transferFrom(address from, address to, uint tokens) public
        returns (bool success);

    event Transfer(address indexed from, address indexed to, uint tokens);
    event Approval(address indexed tokenOwner, address indexed spender, uint
        tokens);
}
```

Listing 2: ERC721 Token Standard Interface

```
pragma solidity ^0.4.20;

contract ERC721 {
    event Transfer(address indexed _from, address indexed _to, uint256 indexed
         _tokenId);
    event Approval(address indexed _owner, address indexed _approved, uint256
        indexed _tokenId);
    event ApprovalForAll(address indexed _owner, address indexed _operator,
        bool _approved);
    function balanceOf(address _owner) external view returns (uint256);
    function ownerOf(uint256 _tokenId) external view returns (address);
    function safeTransferFrom(address _from, address _to, uint256 _tokenId,
        bytes data) external payable;
    function safeTransferFrom(address _from, address _to, uint256 _tokenId)
        external payable;
    function transferFrom(address _from, address _to, uint256 _tokenId)
        external payable;
    function approve(address _approved, uint256 _tokenId) external payable;
    function setApprovalForAll(address _operator, bool _approved) external;
    function getApproved(uint256 _tokenId) external view returns (address);
    function isApprovedForAll(address _owner, address _operator) external view
         returns (bool);
}
```