Decentralized Access Control Solution

Ray Smets
rsmets3@gatech.edu
July 25, 2020

***Abstract*—Online identity is in the hands of each service provider (Facebook, Reddit, Twitter, Chase Bank, etc) and users' authentication credentials are used to temporarily access those identities and accounts. If one of those services leaks account credentials, our service identity integrity is ruined. Additionally, if an inside malicious actor wants to impersonate an account, the capabilities for doing so are present. The blind trust users must put in service providers should not be required. Users should be in full control of their online identities.**

With new decentralized technologies that leverage asymmetric cryptography characteristics, users can now control their online identities. Breakthroughs in decentralized computing technology have democratized access to decentralized compute platforms for application developers, capable of supporting significant real-world applications at scale.

***Index Terms*—Decentralized Computing, Asymmetric Cryptography, Access Control, Blockchain Technology**

I. Introduction

Individuals could follow cyber security best practices and never fall victim to a hack - however their service level identities could still be stolen through no fault of their own. This is a problem.

Having been operating a SaaS product offering for two years I have realized that I could impersonate all the users of our system. Thanks to close source code, I could be storing passwords in the clear, logging their credentials upon hitting log in endpoints, or just be using their session tokens. In the case of Twitter, they had an administrative portal that allowed for account modifications which allowed unauthorized individuals access to high-visibility accounts - effectively allowing impersonation [12].

Using modern decentralized computing platforms that leverage asymmetric cryptography characteristics as a basis of identity, one can remove the centralized identity authority held by each service provider. The key pairs for said authentication are created and kept solely by the entities they are being used to represent. No mechanism for service providers to act on one's behalf or allow access to credentials outside of the account secures a user's service identity. The new authentication paradigm will for users to restore their online identity integrity.

II. Motivation

A. Real-World Example: Twitter Hack

On July 15th, 2020 a Twitter hack allowed for tweets to be sent from accounts known to represent public figures such as Joe Biden, Elon Musk, Apple Corporation, and others. Malicious actors were able to gain access to all these seemingly disparate accounts because of the centralized authorization mechanisms that are standard for many popular online services today. In reality, when users go to use Twitter and they enter the access credentials - secure password, biometrics, Two Factor Authentication - they are in effect asking for permission to then lease their online identity from the service provider. While a disguise of account ownership exists, it is untrue. The service retains levels of control that would allow them to impersonate or allow others to obtain the credentials to impersonate any account holder.

In this case, access to an internal tool was leveraged to manipulate accounts in a way that allowed for credential resetting and possibly credential-stealing [9]. This tool that was exploited was available to 1000 Twitter employees at the time of the hack [12]. This is clearly in violation of the least privilege security principle. Twitter users expect that their account security is held to a high standard, however this hack shows that even high profile companies fail to provide integrity for their user's identities. The accounts that were affected appeared to be at no fault of the account owners, but was a failure of the service provider, Twitter.

B. Personal Experience

Prior to the mentioned Twitter hack I had come to the realization that the power I have over the proprietary access control solution I work on afforded me the ability to impersonate our users. Furthermore, I had the ability to manipulate application state. Both actions can be taken with no customer or user awareness. The customers must trust that I act with integrity and operate with principles of least privilege to ensure their service level intentites are kept secure. I felt that this need for implicit trust in the service provider in regard to identity and application state should be addressed.

III. New Application Deployment Platform: Blockchain

Blockchain as a compute technology is still relatively nascent. It is most popular for supporting cryptocurrencies and thus the public perception of it is that it is used for value store and transfer. Fundamental characteristics that are ubiquitous across the different types of blockchains: transactions are immutable and permanent, asymmetric cryptography is the basis of identity, and the network is distributed using consensus protocols to keep state. Participants that operate the independent nodes of the network are incentivized to do so.

A. Identity

Identity on blockchain networks leverages asymmetric cryptography characteristics allowing for digital signatures as proof of private key possession in a secure manner.

IV. General Consensus Protocols

A. Proof of Work (PoW)

This consensus protocol is maintained by a network of mining nodes that make use of specialized hardware, ASICS, to solve complex cryptographic problems [3]. Blocks are mined in correlation to the complexity of the problem. Miners are only able to add a new block if they can determine the solution. Hence the name is appropriate, the consensus protocol requires computational work for the miner to be rewarded for handling the next block, which is incentivized to offset the costs required to do the work to ensure the health of the network in general. Blockchain technologies such as Bitcoin and Etherium use this consensus protocol which results in low transaction throughput capabilities. Furthermore, the incentives the miners receive are in part made up of a sum of the fees that transactions must pay for resolution. This consensus protocol is flawed for a decentralized compute platform due to the costs pushed to the individuals making the transactions and the low transaction throughput.

B. Proof of Stake (PoS)

Instead of relying on external investments, power consumption, and hardware, of the PoW protocol, Proof of Stake is secured through internal investments. The chance of being the node selected as the block validator is a function of the amount of the underlying blockchain token of which you possess [3]. The health of the network is through all nodes having a significant stake in the health of the network thus ensuring that they are capable of handling the block validation process if called upon is in their best interests. Furthermore, tokens are spawned as an added incentive to the chosen node.

C. Limitations of Public Blockchain Technology

The ideal blockchain characteristics show some promise for a compute platform however for real-world technical reasons these public blockchain networks have not gained meaningful traction supporting applications. Existing blockchain platforms are burdened by large fees and limited computational capacity that prevent widespread blockchain adoption [4].

These limitations can be attributed to why the public perception of blockchain technology is that it is only useful as a way to store and transfer value. However, the latest public blockchains are beginning to show promise as a generalized computing platform.

V. EOS Platform

The EOS blockchain network was officially launched June 2018. The EOS.IO software is designed with proven concepts and best practices representing fundamental advancements in blockchain technology. The software is part of a holistic blueprint for a globally scalable blockchain in which decentralized applications can be easily deployed and governed [1].

EOS uses P-256 Elliptic Curve Cryptography, ECC, keys for authentication, and identity. Identity can be proven and transactions signed via the Elliptic Curve Digital Signature Algorithm, ECDSA. Leveraging fundamental properties of cryptography the digital signature mechanism allows for private key possession to be proven while never needing to be revealed.

VI. Delegated Proof of Stake Consensus Mechanism

EOS uses a Delegate Proof of Stake, DPoS, consensus mechanism that allows for much faster transaction consensus resolution and increased overall throughput in comparison to generalized PoW and PoS blockchains. For example Etherium a PoW blockchain can handle 12-15 transactions per second while EOS can handle 4000 [17]. Tezos, a popular general PoS blockchain can only handle 40 transactions per second [19]. The way EOS touts orders of magnitude better throughput is through a slightly more centralized active network. Stakeholders participate in elections, delegating their tokens to entities to handle the propagation of blocks. The top twenty-one vote recipients are the chosen active nodes and the remaining top 101 block producers are considered on standby in order of their votes received [16]. Just like in generalized PoS these top 122 nodes are incentivized via spawned token rewards.

Other factors that also contribute to the vast scalability improvements of the EOS blockchain over others to date include customized EOS VMs and leveraging Web Assembly as the compiled smart contract byte code [4]. All these elements combined are what positions EOS as the first public chain to be able to support applications at scale.

VII. EOS Network Resources

In the EOS network, three resources are scarce: RAM, CPU, and NETwork.

RAM is the resource on the network that provides permanent storage and is the most expensive part of the distributed compute platform. It needs to be purchased using the blockchain's native token, EOS. At the time of writing the cost of 1KB of RAM is .05 EOS or $0.13 [5]. Different economic models can be employed while operating a decentralized app, the most common to stoke adoption is for the contract developer and owner to be the one responsible for purchasing the RAM for the app, this can be loosely referred to as the "airdrop" model. Conversely, a dapp developer could ask for the users to pay for RAM to interface and persist data in relation to the dapp, this model can loosely be referred to as "airgrab" [16].

Regardless of who is providing the resources for the on-chain permanent storage, users can then interact with the decentralized app, via transactions of a smart contract for free. This is based on the staking mechanism that users must participate in order to have access to the CPU and NETwork. CPU is the resource that is used for performing transactions.

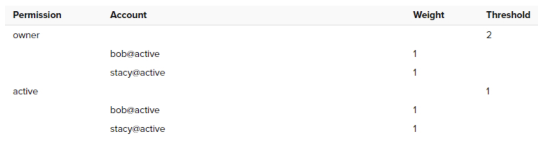
The CPU resource in conjunction with the NETwork is what equates to a user's overall bandwidth for making on-chain smart contract requests and transactions [13].

NET is a space-denominated resource measuring what share of a blocks' network representation can be used to store your transactions as they transmit on the P2P layer [11]. CPU is a time-denominated resource that measures the amount of time an EOS BP, block producer, should dedicate to transactions from your account [11].

Block producers mentioned in the description above are the main nodes of the network where the transactions are physically using compute resources to take place using the DPoS consensus protocol. Tokens staked for NET and CPU resources can be recouped when no longer needed for submitting transactions to the network and RAM can be sold when no longer requiring permanent storage as well [6]. These are the mechanisms that allow for EOS to be relatively cost-free to its users. This distributed compute platform where requests can be made freely is a defining characteristic of EOS [10].

VIII. Account Permissioning

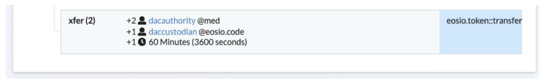
EOS native account permissioning allows for standard enforced least privledge security principles as well as fine-grained hierarchical permission management [8]. The permission management at the account level is what enforces multi-signature transactions.



Permission	Account	Weight	Threshold
owner			2
	bob@active	1	
	stacy@active	1	
active			1
	bob@active	1	
	stacy@active	1	

Fig. 1. Simple multi-signature account configuration.

The image in Figure 1 serves as a visual aid in understanding. While Bob and Stacey are independent accounts their various native (owner and active) account permissions can be used as permissioning for a shared multi-signature account, possibly called "Bob-Stacey". This is a simplistic example however using levels of permission delegation can quickly become complex.



xfer (2)	+2 dacauthority @med +1 daccustodian @eosio.code +1 60 Minutes (3600 seconds)	eosio.token::transfer

Fig. 2. More detailed account permissioning example.

In Figure 2 one can see another alternative permission scheme. Custom permission called 'xfer' requires a threshold of two at-will entities to attempt to meet that threshold. The @eosio.code permission is representative of any smart contract associated with the account, in this case, the smart contract deployed to the daccustodian account. As one can see EOS also allows for time delays as a form of permissions. In this case the code has 'xfer' permissions after a 60 minute time delay.

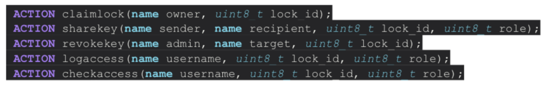
One can view the Appendix's Figure 7 for a visual aid understanding of the EOS accounts in general and for example permissioning configurations and uses. However, the flexibility of permissioning at the EOS account level means that configuring multi verification procedures and entities is easily achievable.

IX. Solution Realized

Leveraging the advantages of blockchain's identity management via asymmetric cryptography, immutable transactions, and the scalability improvements of EOS I was able to build a first of its kind decentralized access control solution that was unencumbered by its decentralized, blockchain deployment platform. The application performance in comparison to the traditional cloud access control solution I work on is comparable, both allowing for access-list mutation and access logs to persist within milliseconds. However, this solution has the security guarantees of a trustless implementation. The operators of this decentralized access control solution can not impersonate one of the user's identity as the solution does not manage session, accounts, or account credentials. Identity verification for key sharing and the other actions is all done securely through digital signatures allowing for a user's private key to remain solely in their possession. Furthermore, all application state is kept on the blockchain itself meaning the data layer can not be mutated in any authorized way.

X. Implementation

The result implementation is a smart contract deployed to the account myeosacldapp on the EOS mainnet. The smart contract exposes five actions that can be leveraged similar to API endpoints and resolve as immutable transactions on the EOS blockchain.



```
ACTION claimlock(name owner, uint8_t lock_id);
ACTION sharekey(name sender, name recipient, uint8_t lock_id, uint8_t role);
ACTION revokekey(name admin, name target, uint8_t lock_id);
ACTION logaccess(name username, uint8_t lock_id, uint8_t role);
ACTION checkaccess(name username, uint8_t lock_id, uint8_t role);
```

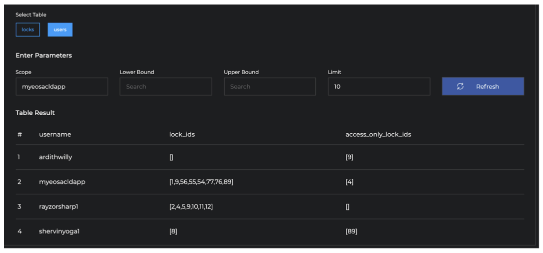
Fig. 3. Application Binary Interface, ABI

The smart contract can be interfaced with using a custom web app that I built to facilitate usage. For a view of the post-authenticated web app user interface one can view in the Appendix's Figure 8.

Using standard EOS app practices the web app leverages a wallet called Scatter for authentication. As long as an EOS account holder keeps their keys secure within a Scatter wallet they can interface with my smart contract via my web app. However, using my web app is not the only way to leverage my smart contract. Because the smart contract can be leveraged via any EOS mainnet node API one can interface with it programmatically or via web apps referred to block explorers. These are services that allow for chain state and transactions to be observed. Some prominent EOS block explores that can be used to interface with my smart contract and view the on-chain application are:

- EOS Authority

- EOSpark
- Bloks.io Account
- EOSX



Scope	Lower Bound	Upper Bound	Limit
myeosacldapp	Search	Search	10

Table Result

#	username	lock_ids	access_only_lock_ids
1	ardithwilly	[]	[9]
2	myeosacldapp	[1,9,56,55,54,77,76,89]	[4]
3	rayzorsharp1	[2,4,5,9,10,11,12]	[]
4	shervinyoga1	[8]	[89]

Fig. 4. The on-chain state of the myeosacldapp application via EOS Authority

As one can see in Figure 4 the state of the application is publicly visible. Additionally one can see the role-based properties of my access control solution. Looking at the state of the application the account shervinyoga1 does not have privileges to share access to lockId 89. They only have the right to leverage the key. These rights can be confirmed programmatically using the check access contract action or the native eos blockchain table struct queries. While on the other hand, the account shervinyoga1 has admin rights to lockId 8 which affords them the ability for key management, sharing and revoking, for that key.

One can also view key chains from a lock's perspective as shown in Figure 5.

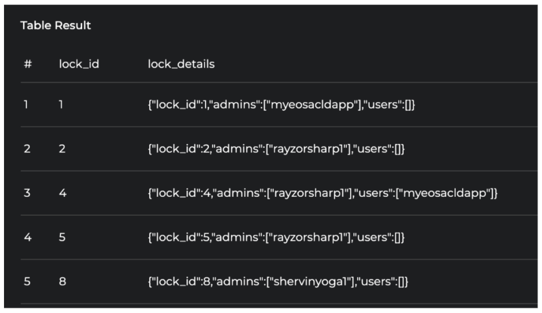


Table Result

#	lock_id	lock_details
1	1	{"lock_id":1,"admins":["myeosacldapp"],"users":[]}
2	2	{"lock_id":2,"admins":["rayzorsharp1"],"users":[]}
3	4	{"lock_id":4,"admins":["rayzorsharp1"],"users":["myeosacldapp"]}
4	5	{"lock_id":5,"admins":["rayzorsharp1"],"users":[]}
5	8	{"lock_id":8,"admins":["shervinyoga1"],"users":[]}

Fig. 5. Key chain view by lock IDs via EOS Authority

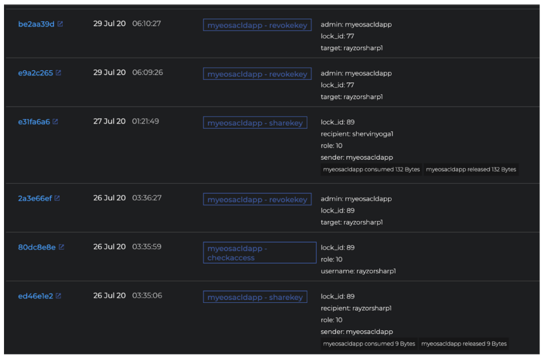
Lastly, the access history and access list mutations can be seen on the immutable ledger of the blockchain in Figure 6.

The advantage of using the custom web is that it facilitates showing one's own keychain and formatting some of the contract information in more human readable format, for example, the role attribute which is an enum with values 10 and 20. The web user interface strictly shows the human readable resolved values of "user" and "admin".

XI. Evaluation

A. Usability

Leveraging information from my professional responsibilities I was able to model a generic customers access control usage. On average customers have two locks and four hundred and thirty persons. Analyzing my smart contract source code



be2aa39d	29 Jul 20	06:10:27	myeosacldapp - revokekey	admin: myeosacldapp lock_id: 77 target: rayzorsharp1
e9a2c265	29 Jul 20	06:09:26	myeosacldapp - revokekey	admin: myeosacldapp lock_id: 77 target: rayzorsharp1
e31fa6a6	27 Jul 20	01:21:49	myeosacldapp - sharekey	lock_id: 89 recipient: shervinyoga1 role: 10 sender: myeosacldapp myeosacldapp consumed 132 Bytes myeosacldapp released 132 Bytes
2a3e66ef	26 Jul 20	03:36:27	myeosacldapp - revokekey	admin: myeosacldapp lock_id: 89 target: rayzorsharp1
80dc8e8e	26 Jul 20	03:35:59	myeosacldapp - checkaccess	lock_id: 89 role: 10 username: rayzorsharp1
ed46e1e2	26 Jul 20	03:35:06	myeosacldapp - sharekey	lock_id: 89 recipient: rayzorsharp1 role: 10 sender: myeosacldapp myeosacldapp consumed 9 Bytes myeosacldapp released 9 Bytes

Fig. 6. Immutable blockchain transactions as they related to myeosacldapp

I determined that a key share requires 10 bytes of RAM, on chain permanent storage. In order to support this average customer usage scenario roughly 8.6KB would be required. Given the price of RAM at time of writing this would cost the customer $1.30 of permanent storage costs. As stated, this cost can be recuperated if the need for permanent storage, which is a function of key shares, decreases.

B. EOS Network Security

The security of my smart contract lies largely in the security and health of the EOS network as a whole. This includes the health and independence of each block producer and the EOS virtual machines of which Web Assembly, the compiled form of the smart contracts, is executed. Thanks to the distributed nature of the network the availability characteristics are high. Additionally, the independent nature of each block producer creates a level of redundancy if one or, an unlikely event, of a few of the nodes are compromised.

C. Smart Contract Security

Following the security research paper Security Analysis of EOSIO Smart Contracts I performed bytecode analysis techniques on my smart contract compiled web assmelby, WASM, files. This enabled me to conduct a security evaluation of my smart contract as it would be executed on chain. While the static analysis tool the researchers created and used for their research was unavailable online I did find a suitable tool called Octopus for my analysis [14]. It allowed me to create a disassembled assembly code view and a control flow graph which can be found in the Appendix's Figure 9 and 10 respectively. The conclusion was that my source code did not have the vulnerabilities discovered in the paper, which were mainly exploits of gambling transfer of value applications dealing with the native EOS token [7]. Additionally, a large percentage of the attacks found to be exploited via poor action permissioning at the smart contract level. Thanks to my access control solution not dealing with transfer of value and ensuring all actions have proper permissioning my bytecode analysis was evidence of the security of the implementation of smart contract at the source code level.

D. Wallet Security

A critical element of the asymmetric cryptography authentication and identity paradigm is the confidentiality of the private key. Leveraging wallets to keep private keys secure is the best way to ensure this. Furthemore wallets expose funtionlaity to application delverops allowing for a standardized way to request transactions to be signed securely enabling a level of trustlessness between the users and the decentralized applications. However significant trust must be given to the wallet in which the individual uses. If that wallet has malicious code the private key can very easily be leaked and the whole account's identity compromised. Furthermore the compute environment of which the wallet runs must also be relatively secure.

For these reasons the use of hardware wallets is suggested for storing the most sensitive keys. As stated EOS accounts come with hierarchical permissioning with account recovery in mind if needed. Leveraging a hardware wallet to store the owner key pair's private key is the most secure configuration. However for convenience keeping the active key pair's private key in a wallet that is readily available for signing makes the most sense. In this way EOS allows for accounts to be configured in a way that is secure yet still maintains a high level of usability.

For my solution I built a sole integration with the Scatter wallet as it is considered the most secure within the EOS community. With that said, I did not conduct a thorough security evaluation myself. However researching the qualities of Scatter it has some interesting added level of reputation and identity to warn users if they are prompted to sign a transaction from an account that has a low integrity rating [18]. These ratings are determined in a crowd sourced manner. Thanks to its own highly regarded reputation and integrity insuring benefits for its users Scatter is the most popular wallet choice for EOS account holders and application developers alike.

XII. Future Work

I feel there are several key areas for usability and valued added improvements to be made. Currently the objects being secured are just eight bit unsigned integers. Supporting access control to something along the lines of SHA-256 files hashes would make for a greater value add. However the advantages of securing integer lock ID values can best be realized if I were to configure my proprietary access control solution mobile apps to speak to an EOS node, leveraging this smart contract as the access list of truth. It would allow for a more holistic and result drive view of this access control solution. Ideally I would very much like to productize this solution, allowing customers a more secure choice of access control.

However, the need for users to have an EOS account and keys in a Scatter wallet makes for user onboarding very steep. With a truly productized version of this decentralized app I could incur revenue to cover the costs of EOS account creation, which requires RAM. As well as the costs of the RAM, on chain permanent storage, operating costs of the application could be fronted with the revenue. Both elements would drastically lower the barrier for customer adoption. Given the very low operational costs I feel there is a great deal of monetary margin to be made operating said solution.

Furthermore there are very many new and useful solutions to support truly productized blockchain applications. May of them deal with blockchain data history and federated accounts between blockchains [2]. Both areas I now understand much more clearly as I can see a very real need for solutions in both areas.

XIII. Conclusion

I am very pleased with the resultant decentralized access control solution built on EOS. The aims that I set to address regarding implementing a trustless access control solution have been met. Users can leverage this solution without concern of the service leaking their service identity while also having the confidence that application state can not be manipulated except with those with proper privileges. Lastly the access history is also immutable allowing for an undisputed accurate audit log capabilities.

Having the opportunity afford by this project to spend a considerable amount of time in the still extremely nascent blockchain technology space I can say with a high degree of confidence that the areas that are best suited for the compute technology are IoT, social media, and artificial intelligence. Furthermore, EOS is certainly the best suited public blockchain for apps at scale today. However, it still has some way to go before it can support anything close to traditional service providers. It is clear that for the foreseeable future blockchain adaptation will be a function of its capabilities with slow but steady improvements. I look forward to the security benefits it will bring to users in mass.

References

[1] Ameer Rosic, *What is EOS Blockchain: Beginners Guide*, Retrieved July 30, 2020, from https://blockgeeks.com/guides/eos-blockchain/

[2] A. Banafa, *Ten Trends of Blockchain in 2020*, Retrieved July 27, 2020, from https://www.bbvaopenmind.com/en/economy/finance/ten-trends-of-blockchain-in-2020/

[3] Binance Academy, *Delegated Proof of Stake Explained*, Retrieved July 30, 2020, from https://academy.binance.com/blockchain/delegated-proof-of-stake-explained

[4] Eosio, *EOSIO/Documentation*, Retrieved July 30, 2020, from https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md

[5] Fastest EOS Block Explorer, Retrieved July 30, 2020, from https://www.eosx.io/tools/ram/buy

[6] D. Floyd, *RAM It All: Rising Costs Are Turning EOS Into a Crypto Coder's Nightmare*, Retrieved July 30, 2020, from https://www.coindesk.com/ram-it-all-rising-costs-are-turning-eos-into-a-crypto-coders-nightmare

[7] N. He et al., *Security Analysis of EOSIO Smart Contracts*, Retrieved 2020, from https://arxiv.org/pdf/2003.06568.pdf

[8] Y. Instructor, Y. Slenter, *EOS Permission Management*, Retrieved July 30, 2020, from https://www.eosbootcamp.com/lesson/multi-signature-accounts-2/

[9] B. Krebs, *Who's Behind Wednesday's Epic Twitter Hack?*, Retrieved July 27, 2020, from https://krebsonsecurity.com/2020/07/whos-behind-wednesdays-epic-twitter-hack/

[10] LiquidApps, *Abundant Decentralized Storage on EOSIO*, Retrieved July 27, 2020, from https://medium.com/the-liquidapps-blog/abundant-decentralized-storage-on-eosio-a1fb07c63618

[11] LiquidEOS, *CPU, NET & RAM - The raw materials of the EOS economy*, Retrieved July 27, 2020, from https://medium.com/@liquideos/cpu-net-ram-the-raw-materials-of-the-eos-economy-c4f85022fae

[12] J. Menn, *Exclusive: More than 1,000 people at Twitter had ability to aid hack of accounts*, Retrieved July 27, 2020, from https://www.reuters.com/article/us-twitter-cyber-access-exclusive/exclusive-more-than-1000-people-at-twitter-had-ability-to-aid-hack-of-accounts-id

[13] Polar, *What is CPU? What is RAM? And How Does the EOS blockchain Utilize These Resources?*, Retrieved July 30, 2020, from https://medium.com/@polar_io/what-is-cpu-what-is-ram-and-how-does-the-eos-blockchain-utilize-these-resources

[14] Pventuzelo, *Pventuzelo/octopus*, Retrieved July 27, 2020, from https://github.com/pventuzelo/octopus

[15] R/eos - Any easier explanation for EOS RAM for non-technical people?, Retrieved July 30, 2020, from https://www.reddit.com/r/eos/comments/9amk0h/any_easier_explanation_for_eos_ram_for_non/

[16] K. Samani, *The Definitive Voting Guide For EOS Block Producers*, Retrieved July 30, 2020, from https://www.forbes.com/sites/ksamani/2018/09/18/the-definitive-voting-guide-for-eos-block-producers/

[17] A. Satya Avala, *Why we built our blockchain business on EOS instead of Ethereum*, Retrieved July 27, 2020, from https://venturebeat.com/2019/04/13/why-we-built-our-blockchain-business-on-eos-instead-of-ethereum/

[18] Scatter, *RIDL 1.0*, Retrieved July 30, 2020, from https://docs.google.com/document/d/1cDcERQUAA3W21XtJhHlqfL2LauiELmOE_b5J2Q3acJg/edit

[19] Tezos (XTZ), Retrieved July 30, 2020, from https://blockfyre.com/tezos-xtz/

APPENDIX

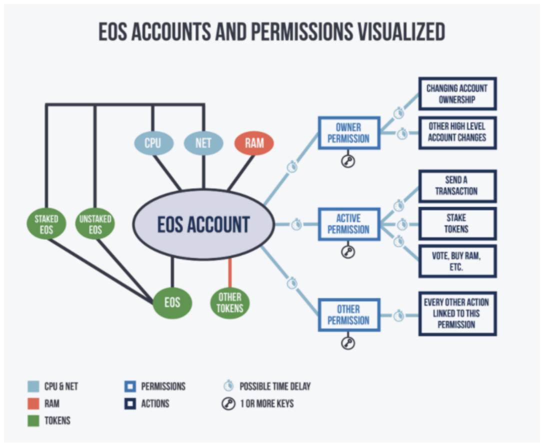
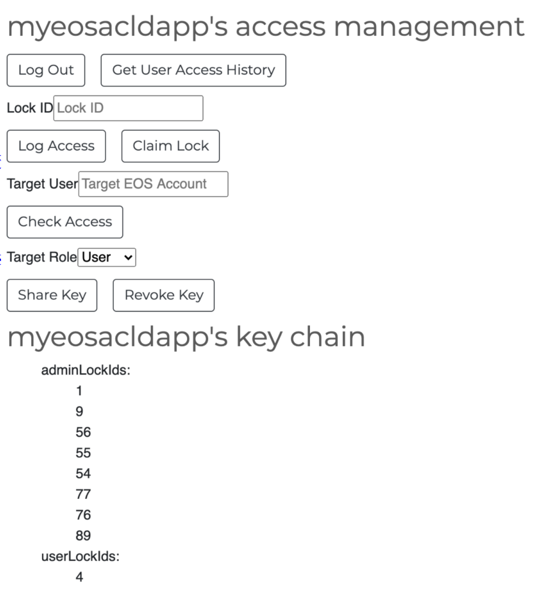


Fig. 7. EOS account permission visualization



myeosacldapp's access management

Log Out | Get User Access History

Lock ID Lock ID

Log Access | Claim Lock

Target User Target EOS Account

Check Access

Target Role User

Share Key | Revoke Key

myeosacldapp's key chain

adminLockIds:
1
9
56
55
54
77
76
89
userLockIds:
4

Fig. 8. View of decentralized app interface from the custom web application

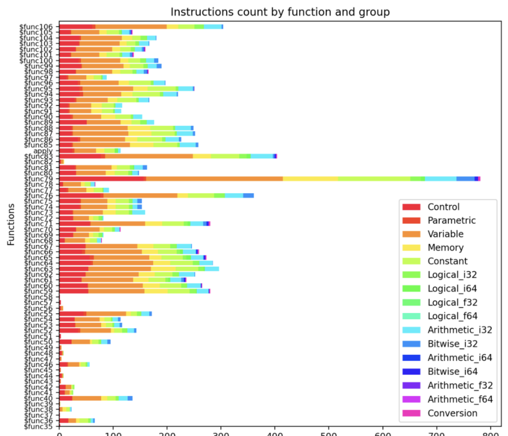


Fig. 9. Octopus byte code instruction analysis of myeosacldapp

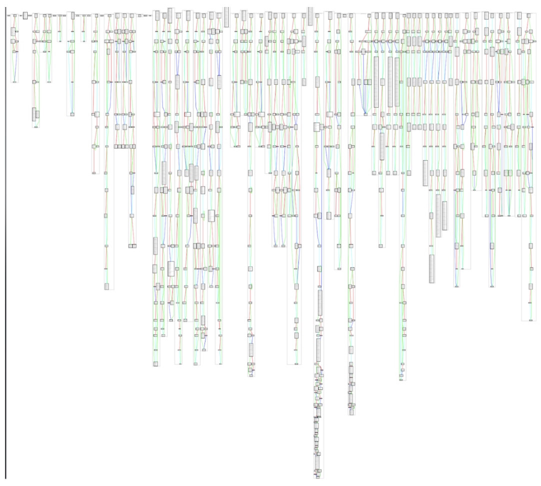


Fig. 10. Octopus byte code flow diagram of myeosacldapp