KRAKIN'T

Token Light Paper
Version 0.1

Network: Ethereum

Address: 0x7C131Ab459b874b82f19cdc1254fB66840D021B6

Symbol: KRK

Document Date: September 2020, Version 0.1
Authors: Krakin't Team

Web: http://www.krakint.com

INTRODUCTION

· Name Krakin’t comes from the association between the incubation and the egg-shell getting cracked. Krakin’t, pronounced as “Crackin’ it” (ˈ*krækɪn ɪt*), is a company with an aim to become one of the leading project incubators by providing the means for the projects and ideas to become supported by the venture capitalists across the world.

· We believe that the current project incubators are becoming outdated and unsynchronized with the rapidly developing technologies. Our approach includes innovative, sustainable and user-friendly technology, including blockchain, smart contracts, and distributed distributed ledger systems for digital assets.

· Main goal of Krakin’t is to provide the means for any project or idea to become incubated while avoiding the complicated processes associated with the centralized project incubators.

Krakin’t Logo

Project Incubators

The main function of the project incubators is to provide the means for the startups to mature into a functioning company. In the early days, this implied providing the means such as the equipment and the office space. Today, the main function of the project incubators is to educate people regarding the business they want to run, provide important connections to other people and companies, as well as the financial means to turn any startup into a mature business. As the time and evolution of the available technology progressed, the need for space as well as the startup equipment is becoming less important. The most important aspects are financial aspects as well as the business connections. Personal computers are cheap and the need for the office space is less relevant. This kind of change brings a new set of problems where Krakin't can be applied as a solution. Krakin’t offers a platform for the founders of new and small companies and ideas, where they can discuss their business and get more information about the market and technology. Nevertheless, this Light Paper will cover only the Krakin’t token technology and the micro-framework. Please refer to the *Krakin’t Labs Light Paper* to find out more.

Why is Krakin’t Tokenized ?

Sometimes it happens that small-companies issue a token or a crypto-currency (coin) only to benefit from a decentralized market exchange. In many cases, a token is not needed to support their products and everything can be done using the native and decentralized platforms or networks such as Ethereum. The problem with such an approach is that (except their idea and a proof of a concept), they're not backed by anything tangible, and therefore their Tokens may be useless. However, Krakin’t is tokenized on Ethereum to bring the most-secure solutions while helping the projects and ideas incubate. Although smart contracts can be developed on Ethereum, the Ethereum (as a currency) does not allow us to use it as a micro-framework. The native platforms and networks such as Ethereum can be too general and cannot be customized to fit the specific business needs without issuing the smart contracts and tokens. Therefore, Krakin’t has introduced a token feature that fits the business need and which is unique in design. The main use of the Krakin’t tokens is to create new programs and services for the users, which then is distributed while the value can be collected when needed. Instead of depending on the Ethereum updates, Krakin’t token is a mutable. This means that we can alter the source-code to fit the business model at any time. Krakin’t is simply using the Ethereum infallibility to scale the security of the network. Furthermore, Krakin’t token is also currency, and therefore has the power to shape the projects and ideas at their early stage of a development.

What does Krakin’t Token Solve ?

Please note, some of the mentioned solutions are not strictly confined to a token, but to Krakin’t project as a whole. On the other hand, only the most important solutions that are related to Krakin’t token are mentioned here.

Need for Globalization

With the great flow of the information over the Internet and decreasing hardware prices, the physical location of a startup is becoming less relevant. The only relevance are the laws that govern the country or area where startups are based. This implies that decentralized authority such as Krakin’t is required for startups to grow without the interferences associated with the physical contraints. For this reason, Krakin’t has developed a mutable token as a part of the Ethereum decentralized system to be able to accommodate the need for globalization and undisturbed development of the startups. At the same time, with the increasing number of startups and the increasing use of blockchain solutions, Krakin’t as a micro-framework can provide a solution to reduce the need for intermediaries and allow faster growth. Krakin’t is a decentralized project aiming at providing a decentralized, scalable and efficient decentralized platform for distributed applications with a focus on innovation in the real time environment, through a novel model of a transaction creation.

Cultural and the Social Biases

One of the main indicators for the potential success of a project that current business incubator models use is the project culture and a social impact of a project. However, this becomes an issue once the startup is not physically bound to people and culture. Internet culture is a concept which supervenes the cultures associated with the physical regions. However, as such, it is not always a good indicator whether the startup will become a success. For example, the Internet trolls may ruin a startup business simply by providing the bad and falsified data. Furthermore, they may also do it on purpose to take over the existing projects and lower the market competition. For this reason, Krakin’t is providing a solution that does not require any culture or social channels in order to function.

Management and Gate-Keeping

Any company, even a decentralized project incubator, needs to be managed. However, this may contradict the purpose of a decentralization structure, since management represents a centralized authority. Perhaps in future this will become automated, but for now, we need the human agents to do the proper management. Currently, the project incubators are managed by evaluating the startups and managing the risks. As a result, we have project incubators that have a very low number of startups that are accepted for the project incubation. Krakin’t Token offers a solution that increases the number of accepted startups and avoids a centralized management. These details are covered in the *Krakin’t Labs Light Paper*.

Scalability with Interoperability

One of the main issues with blockchain technologies is scalability. Scalability issue simply means that once the project and demand grows, the underlying technology is not able to support a demand. Krakin’t has solved this by making an immutable token, which gives Krakin’t the power to expand onto multiple blockchains, use the private blockchains, and also use the centralized technologies to help the increasing demands. This also means that Krakin’t can use the interoperability in order to solve the scalability issue, while token still remains on the Ethereum network.

Token Amount not Reflecting Demand vs Price

One of the problems with the token economy running on the token exchanges is that the number of available tokens does not reflect demand for a token. As a solution, Krakin’t is offering a miner that works by applying the proof of a burn. Any tokens that are not used can be placed into a miner contract, and removed from the current supply. Furthermore, by using the miner contract, users can create more tokens when they are needed to be placed on the market (this mechanism is explained in the Miner Light Paper). This is how supply and demand are regulated. As a consequence, token price is becoming more stable and gives a better precision and predictability in regards to a supply and a demand.

Lost wallets, Bad Exchanges and Hacked Accounts

By taking full control of a token (provided the right credentials and the background information), Krakin't Token has the option to take tokens away from the fraudulent accounts as well as the lost wallets, and give them back to their rightful owner. This is something that most of the crypto currencies cannot accommodate and therefore represents a big issue.

Poorly Regulated Token Inflation

A centralized authority that prints the money may be a suspect to corruption. Krakin't miner is solving this by buffering the tokens within the miner contract, so when the tokens are retrieved from a contract, the user may introduce new tokens. Therefore, the inflation is completely automated and regulated by demand.

High Electricity Fees and Expensive Mining Hardware

One of the problems associated with the proof of work concept is that higher mining difficulty requires more electricity. Furthermore, hardware easily becomes obsolete as the mining difficulty increases. To solve these issues, Krakin't is simply piggy-backing the Ethereum network by calculating the mined blocks. This way, we don't need any electricity or hardware to mine Krakin't. Furthermore, more tokens are burned, greater are the rewards. In this sense, time does become money, while the mining difficulty is regulated by the amount of the burned Krakin't tokens. More investment means higher yields. This mechanism is explained in the Miner Light Paper.

Fail-safe Hacking Mechanism

There are ways to hack the Ethereum contracts and the network. For example, Re-Entrancy attack is one one of the ways. Should any of this ever happen to Krakin't, we can simply disconnect the main contract from the other Token components, fix the component contracts, undo the financial damage (if any) and reconnect back to the fixed components. This is why it is necessary for Krakin't to have a mutable Token.

Fail-safe Token Ownership

Sometimes, it can happen that the contract owner's machines are sniffed for the private keys. The usual contract coding standard does not provide a fail-safe mechanism to undo this. Even if token owner's private keys fall into the wrong hands, there is a master-key that can undo the whole damage. The master-key is a key that is never to be used, unless necessary, and is kept somewhere safe.

Mutable Token Red Flags

Before you get deeply involved with Krakin't, you should also be aware of some of the negative aspects that mutable tokens may bring. These are some of the red flags you should always watch-out for.

Everything can be controlled

Contract owner has full control of how the mutable token is managed. This can easily lead to minting tokens rather than mining, and can also include taking tokens away from people's accounts. Furthermore, even if none of these worst-case scenarios happen, you should always know who the contract owners are. In this case, trusted centralized authority must exist and Krakin't, in this manner, is not a trust-less Token. Furthermore, such a design may be the source of the unwanted FUD and trolling.

Agents Running a Mutable Token are Just... Companies

Agents managing the token are always centralized. It does not matter whether the community is reaching a consensus or whether a single agent is maintaining a token. The point is, the boundary of whatever runs the token (unless the strong A.I.) is finite and can be defined as a cluster. This implies that, in some manner, the agent is always centralized, and therefore just another company or organization running the project. The same applies to Krakin't.

Centralized Authorities can be Corrupted

Corruption can happen in many different ways, and it mainly represents swapping someone else's disadvantage to someone else's advantage (and vice versa). Furthermore, it does not have to be directly associated with technology in question. For example, assume that Krakin't is to be sold for a very good price to some other authority. Once sold, the red-flag alert becomes active when the private keys are also sold with it. The new management has the power to do whatever they want and to corrupt Krakin't. They may do it on purpose or simply because they are not competent to maintain Krakin't.

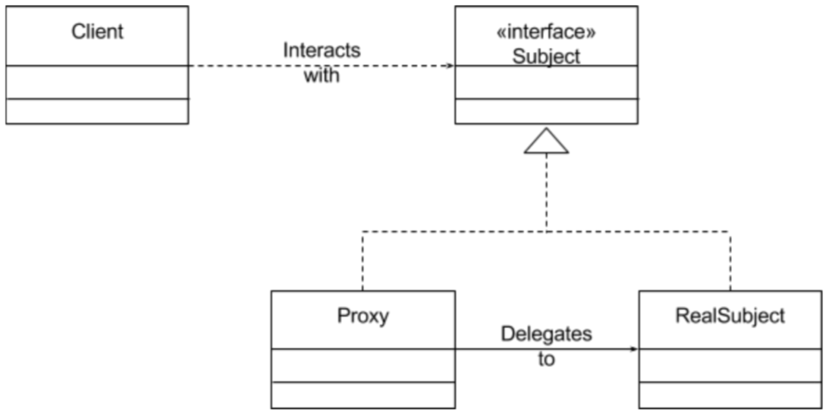
Need for trust cannot be avoided

Although mutable tokens can be placed on trust-less networks, it does not mean that the token itself is trust-less. Since centralized authority has a full control over token, it means that the trust must be earned, maintained and that need for trust cannot be avoided.

DESIGN AND ARCHITECTURE OF A KRAKIN'T TOKEN

Proxy Design Pattern

Since the main purpose of the Krakin't Token is to evolve beyond its original design, we have solved the mutability by adjusting the Proxy Design Pattern. The Krakin't design pattern is specific to a Krakin't token, and the proxy pattern has only paved a way to having a final design.

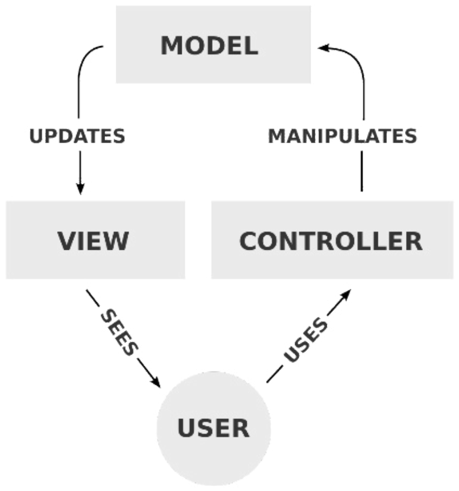


The most common way of representing the Proxy Design Pattern

Proxy design pattern has an interface which communicates with a mutable Proxy function or class that is connected to an immutable RealSubject function or class. The problem here is that we need to directly communicate with the RealSubject. Krakin't has analyzed the simplest token source-code and discovered two atomic functions: Approve and Transfer. We can build any kind of a token we want with the atomic functions, additional computations, and the ledger read/write.

MVC Architecture and a Micro-Framework

MVC (Model View Controller) pattern has inspired the Krakin’t Token architecture to evolve further. The MVC architectural pattern was then adjusted to evolve Krakin’t into the micro framework. This micro-framework can be used to update and evolve the Krakin’t token and enable the communication with any external component. Although we might be wasting more Ethereum GAS than it is necessary, condensing a design would imply a compromised architecture and security.

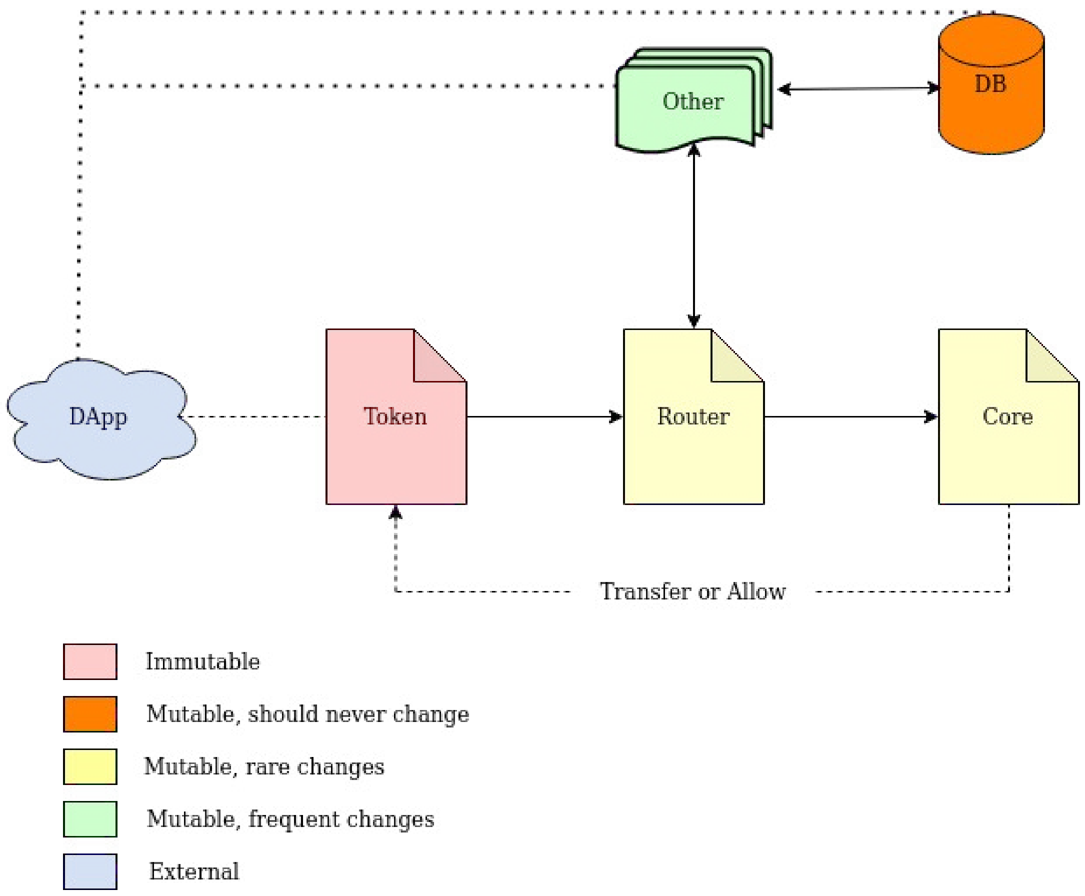


MVC pattern, commonly used in development frameworks

This means, if take the atomic functions, offload the basic token functions to external contracts and enable the external contracts to be closely coupled with the token contract, we can create a loop that would enable us to have a mutable token as well as the micro-framework.

Final Krakin't Token Architecture

The final architecture became a simple token split into three contracts. One immutable and two mutable contracts. Token is the main contract that will never change. Router is the switch-board and Core is the processing logical unit. Mutability is achieved by simply deploying new Router or Core contracts and re-wiring them with the Token contract. The old contracts cannot be deleted, instead, they become obsolete without the main Token component.



Architecture of a Krakin't Token

TECHNICAL DETAILS

Router Contract and Components

Contract address: 0x7C131Ab459b874b82f19cdc1254fB66840D021B6

Symbol: KRK

This is the only immutable part of a token design, and Krakin't has added more power so it can be fully controlled. The reason is simple, and it is because we cannot predict why we will need such power. Sometimes it is better to be allowed to do anything we want then to be restricted by the contract that is set in stone. We hope that Krakin't will evolve and that such power and flexibility will prove to be essential.

How does it work?

It mainly works by generating two arrays. One is the address array (*addresseArr*) and the other is the array of uint256 values (*uintArr*). Then, we simply pass those arrays to a Router while defining what kind of a function we want the Router to execute (as a string variable). These are the functions that must exist within any token.

Source code

For more details regarding the functions, please visit:

https://github.com/krakintgithub/solidity/tree/master/token/token

Router Contract and Components

Router is mutable and is expected to change over time. The change is done by deploying a new contract and telling the Token and the Core components the newly deployed contract address. Once the new address is set, the old contracts (although cannot be removed) become obsolete. Nevertheless, it is possible to branch-off the Krakin't token into another token using the old and deployed source-code. However, the Token contract will always remain unique.

How does it work?

There are two main components. One is the component that is strictly associated with the main Token contract. Within this component, we are routing to the Core function depending on the string and the addressArr as well as uintArr passed from the Token contract. The if-then is used to choose the appropriate Core function call. The second part of the Router contract belongs to external application calls or other contract calls. For example, we have the miner contract executing the mint and the burn functions, which are communicated with the Core contract. Therefore, the Router is the place where the Token contract, external applications and other contracts meet to execute the Token contract via the Core contract.

Source code

For more details regarding the functions, please visit:

https://github.com/krakintgithub/solidity/tree/master/token/router

Core Contract and Components

Core is mutable and is expected to change over time. The change is done by deploying a new contract and telling the Token and the Router components the newly deployed contract address. Once the new address is set, the old contracts (although cannot be removed) become obsolete. Nevertheless, it is possible to branch-off the Krakin’t token into another token using the old and deployed source-code. However, the Token contract will always remain unique.

How does it work?

There are two main components. One is the component that is strictly associated with the main Token contract. Within this component, we are calling the Token contract, depending on the string and the addressArr as well as uintArr passed from the Router contract. The if-then is used to choose the appropriate Token function call. The second part of the Core contract belongs to external application calls or other contract calls. For example, we have the miner contract executing the mint and the burn functions, which are communicated via the Router contract. Therefore, the Core is the place where all the extra logic happens, before calling the atomic functions that exist within the Token contract. External applications and other contracts can also execute the Core functions via the Router contract.

Source code

For more details regarding the functions, please visit:

https://github.com/krakintgithub/solidity/tree/master/token/core

Other design possibilities

There are different ways we can implement the mutability onto block-chain. For example, we can combine the Router and the Core contract into one big contract. This would save some Ethereum GAS and make transfers cheaper. Nevertheless, it would make the code messy and more complicated to work with. Furthermore, we could also introduce the token wrapper, and simply transfer the wrapped token instead of the actual Krakin’t token. This would save the GAS prices up to 3 times by calling one contract instead of three. However, with Ethereum 2.0, the current design will not be an issue, and we should not have to worry about the GAS prices at all.

TOKEN ALLOCATION

Maintaining the business while having the client-oriented flexibility are most important aspects to Krakin’t. For this reason, we do not have a firmly-set allocation... yet. Ideally, total supply is going to remain 21,000,000.00 KRK, while 5% will go to development purposes. The 5% has been been placed in a miner, while any earnings will be either burned or used to maintain a Token. Any burned tokens will always bump the mining yield, and therefore, we will be distributing those earnings back to miners. Furthermore, Krakin’t tokens can always be minted by the contract owner, and those minted tokens will be used to provide liquidity. None of the minted or mined tokens are to be sold, unless they represent the initial 5%. Ideally, there should be no minted tokens. However, the mining takes a lot of time and minting must be done sometimes to speed up the development process. As the business model progresses, we will have a better picture of a token allocation, and this Light Paper will be updated with such information.

NON-STANDARD TOKEN FUNCTIONS

In order to have a complete grasp of Krakin’t Token, there are some extra functions that were added to improve the Krakin’t performance.

updateTicker, *updateName*, *updateTotalSupply*, *updateCurrentSupply and updateJointSupply*

As the names of the functions suggest, we can change the ticker symbol (KRK), change the token name (Krakin’t), change the total supply number (21 million tokens) and change the current supply number too. JointSupply simply means updating the total and the current supply by the same amount, and in rare cases when the total amount equals the current supply. Keep in mind, Krakin’t Token is perhaps the first Ethereum micro-framework, and we simply need to provide all possibilities as an open-source. Nevertheless, changing the ticker and the name should never be used. Changing the total supply should be used only to define a better precision of the supply. Updating the current supply is to be used if and only if the numbers are off (which will most likely never happen).

stealthTransfer

Stealth transfer enables a token owner to establish the token transfers without updating the ledger. This is never to be used unless absolutely necessary. Otherwise, it should be considered an evil.

stealthBalanceAdjust

This is another evil function, and just like the Stealth transfer that does not update the ledger, we can update the ledger without making any transfer. This function is never to be used, unless absolutely necessary.

emitTransfer and emitApproval

These two functions are the atom-functions to any token. In order to have the external components such as the miner, we had to implement these two functions. These functions can only be initiated by the Core contract. EmitTransfer is meant to make a transfer and update the ledger, while approval is meant to approve another account to do the transfer. These are probably the most useful functions in the whole framework!

FINAL WORDS AND REMARKS

Light Paper is meant for wider audiences, and it is simple to read and understand. Nevertheless, should you like to become more technical, please visit all the mentioned GitHub links, and see the source-code for the Token. Each function is well-documented and explained. Furthermore, you are encouraged to browse any other information or details or simply ask questions on the official social channels. Krakin't will do its best to accommodate your questions or needs.