TRIP COIN

WhitePaper V.1.3 del 02/10/2021

ABSTRACT

Trip Coin is a virtual currency that uses an Ehtereum protocol with the Erc20 standard and gives access to new applications, information and new theoretical and practical perspectives applicable in all fields and company projects, automatically signing each action directly in the Ethereum blockchain, without any human validation.

Trip Coins are special coins, the Project foresees the use in the Business field and the use of the same as prizes, participations, economic incentives to the companies that intend to adopt them, the small investment of today could be your capital of tomorrow.

Team

The idea of a bargaining chip was born in 2018, from the mind of Tripicchio Fabrizio Fondatore and creator of the project,
https://www.linkedin.com/in/fabrizio-tripicchio-8a79a960/ when companies were looking for alternative forms of remuneration to traditional money.
Thus creating a team in 2019 for development:

Daniele Nuzzolo (BlockChain Developer) https://www.linkedin.com/in/daniele-nuzzolo-5131b21a7/
Dodaro Fabio (BlockChain Developer) https://www.linkedin.com/in/fabio-dodaro-7b190a33/

developed the ERC20 Trip Coin Token in 2020 on the Ropsten network to ensure its correct functioning,
finally expected until the summer of 2021, following the Hard Fork Berlin and London updates, to place it on the Main Net Ethereum

At the same time, figures for Marketing and Press Relations
joined the Team
Drago Marilena (Social Media Marketing) https://www.linkedin.com/in/marilena-drago-9417846a/
Forestiero Francesco (Press) https://www.linkedin.com/in/francesco-forestiero-7260111b/

for the development of an ERC20 Token on the Ethereum network that could serve the purpose, listed below,

Project

The most frequent cases were:
- company partnerships, which required a considerable economic and bureaucratic commitment from small and large companies, with an immediate cash outlay
- bureaucratic acts (notaries, banks, etc ..) due to legally crystallize the acts, involve a considerable additional expense.
- The remuneration of services and / or services provided, such as bonuses or similar, could only be performed against cash compensation, further disbursement from the company coffers
- Last but not least, corporate holdings involved decision-making power, which is often not applied due to practical issues and subordination in the corporate ranks

Trip Coin is proposed as a solution to all the problems listed

- company shareholdings can be divided according to the number of Trip Coins in their possession, which fully guarantee a share of the company both in liquidity and in percentages
- the bureaucratic deeds are delegated to the Ethereum BlockChain which crystallizes the operations in an immutable way exactly like a notarial deed
- remuneration of bonuses, company bonuses and the like can be paid directly by the company in the form of Trip Coin
- the amount of Trip Coins owned implies an equal decision-making power within the Company which can put, exactly as in a popular vote, decisions of small and great importance to the votes, all the carrying out of such voting process is always guaranteed by the Ethereum Block Chain, without the possibility of fraud or fraud.

With this we propose ourselves not only in the business sector, but in most of the processes and services where these proposals are applicable
Fabrizio Tripicchio (CEO) Trip Coin

USABILITY

Currently for sending, receiving, buying Trip Coin, you just need an Ethereum Wallet,(Metamask, Trust Wallet, Etc..) downloadable for free for Desktop / Mobile It is possible to buy Trip Coin for investment and company capitalization: directly on Smart Contract or on UniSwap
.

LIQUID ASSETS'

Guaranteed by Uniswap through the Pool it is possible to buy, sell, exchange Trip Coin with any other Crypto Currency, the value will be updated automatically based on the exchanges made, when it is listed on centralized Exchanges it will have a higher value than the purchase one, a real investment for the future!

BUSINESS APPLICATIONS

Trip Coin allows the internal development of any business platform that aims to improve performance in the reference company.

GOVERNANCE

In a company system divided into company shares, it is difficult to guarantee privacy and management based on the ownership of company shares, Trip Coins can completely replace a system of company governance, giving all its holders decision-making power based on the capacity .

COMPANIES

A use that is well suited to companies operating in various fields are production bonuses, corporate governance (votes, decisions, etc.), staff remuneration.

TOKEN TRANSFER

Data migration or token transfer and validation is the strength of the protocol adopted by Trip Coin. Any operation takes place in real time and guaranteed by the Ethereum Block Chain, in which each transaction is validated and immutable.

WALLET

To hold Trip Coins, exchange them or sell them, it is enough to have an Ethereum Wallet, nothing else is required.

SAFETY

Written entirely in Solidity and protected with SafeMath, it is a Solidity library aimed at managing and blocking integer overflow attacks on contracts, every operation carried out with this Token is

stored in the immutable BlockChain of Ethereum, thus guaranteeing both anonymity and security of the transaction.

TEAM

Development and Administration Tripicchio Fabrizio
https://www.linkedin.com/in/fabrizio-tripicchio-8a79a960

What Are Mintable Tokens?

Mintable tokens are ERC20-compatible tokens with one added feature: new tokens can be created at any time and added to total supply. Standard ERC20 tokens don't have this feature, which makes them a fixed supply tokens

Minter Role

Minter role is an address that has a special privilege to create new tokens, thus adding them to total supply. There could be multiple Minter role addresses, but in practice, it's usually just one address. The first Minter is the address that deployed the token contract. Current minter can add new minter, renonunce his privilege to be minter, or transfer his right to other address.

Mintable Token Examples

There are plenty of mintable ERC20 tokens out there.
Here are some of the more interesting tokens.
Dai (DAI)
DAI is decentralized stablecoin pegged to $1 USD used in the MakerDAO system. Users can lock their crypto assets (ETH) inside of MakerDAO contract, and in return get DAI tokens. Everytime ETH is locked in CDP (Collateralized Debt Position), new DAI tokens are minted by the MakerDAO contract.
Status (SNT)
Status is an open-source platform which uses it's native SNT token as a mechanism for governance of the Status client, as a utility token to drive push notifications in their messaging app, as well as for curation of user-generated content on their network. The SNT token has a cap on total supply, but a lower current circulating supply. New tokens can be minted by the controler address.
Decentraland (MANA)
Decentraland is a virtual world platform owned by its users. Users create goods and services and trade them using MANA tokens. MANA tokens are mintable and burnable ERC20 tokens.

Minting vs Mining

Minting is not the same as mining. This is a common misunderstanding when people are talking about cryptocurrencies.
Minting tokens is done by sending a transaction that creates new tokens inside of token smart contract. As we have seen in What are mintable tokens, a call to a smart contract function can create unlimited number of tokens, without spending energy.
On the other hand, mining does create new tokens, but is usually limited as per concensus rules of that blockchain, and requires spending energy. Mining also serves other purposes like securing the network and packing new transactions into blocks.

Mintable Tokens in Crowdsales

Mintable tokens are widely used in combination with crowdsales. Crowdsale contracts are used to create an ICO where ERC20 tokens are sold for ETH. Here's the mechanism used for combining mintable tokens with crowdsale:

1. ICO maker deploys mintable token contract, giving him a Minter role
2. ICO maker deploys crowdsale contract
3. ICO maker transfers Minter role to crowdsale contract. Only crowdsale contract can mint new tokens.

Now, the crowdsale begins. Investor comes along and wants to invest in new tokens. He sends ETH to crowdsale contract, contract mints brand new ERC20 tokens at current price and sends them to investor. When crowdsale is finished, no one can mint more ERC20 tokens, not even crowdsale contract, since it's coded that way. Investors can be sure that no one can dilute their share of token holdings.

A Next-Generation Smart Contract and Decentralized Application Platform

Satoshi Nakamoto's development of Bitcoin in 2009 has often been hailed as a radical development in money and currency, being the first example of a digital asset which simultaneously has no backing or intrinsic value and no centralized issuer or controller. However, another - arguably more important - part of the Bitcoin experiment is the underlying blockchain technology as a tool of distributed consensus, and attention is rapidly starting to shift to this other aspect of Bitcoin. Commonly cited alternative applications of blockchain technology include using on-blockchain digital assets to represent custom currencies and financial instruments (colored coins), the ownership of an underlying physical device (smart property), non-fungible assets such as domain names (Namecoin), as well as more complex applications involving having digital assets being directly controlled by a piece of code implementing arbitrary rules (smart contracts) or even blockchain-based decentralized autonomous organizations (DAOs). What Ethereum intends to provide is a blockchain with a built-in fully fledged Turing-complete programming language that can be used to create "contracts" that can be used to encode arbitrary state transition functions, allowing users to create any of the systems described above, as well as many others that we have not yet imagined, simply by writing up the logic in a few lines of code.

Ethereum Accounts

In Ethereum, the state is made up of objects called "accounts", with each account having a 20-byte address and state transitions being direct transfers of value and information between accounts. An Ethereum account contains four fields:

The nonce, a counter used to make sure each transaction can only be processed once
The account's current ether balance
The account's contract code, if present
The account's storage (empty by default)

"Ether" is the main internal crypto-fuel of Ethereum, and is used to pay transaction fees. In general, there are two types of accounts: externally owned accounts, controlled by private keys, and contract accounts, controlled by their contract code. An externally owned account has no code, and one can send messages from an externally owned account by creating and signing a transaction; in a contract account, every time the contract account receives a message its code activates, allowing it to read and write to internal storage and send other messages or create contracts in turn.
Note that "contracts" in Ethereum should not be seen as something that should be "fulfilled" or "complied with"; rather, they are more like "autonomous agents" that live inside of the Ethereum execution environment, always executing a specific piece of code when "poked" by a message or transaction, and having direct control over their own ether balance and their own key/value store to keep track of persistent variables.

Messages and Transactions

The term "transaction" is used in Ethereum to refer to the signed data package that stores a message to be sent from an externally owned account. Transactions contain:

The recipient of the message
A signature identifying the sender
The amount of ether to transfer from the sender to the recipient
An optional data field
A STARTGAS value, representing the maximum number of computational steps the transaction execution is allowed to take
A GASPRICE value, representing the fee the sender pays per computational step

The first three are standard fields expected in any cryptocurrency. The data field has no function by default, but the virtual machine has an opcode which a contract can use to access the data; as an example use case, if a contract is functioning as an on-blockchain domain registration service, then it may wish to interpret the data being passed to it as containing two "fields", the first field being a domain to register and the second field being the IP address to register it to. The contract would read these values from the message data and appropriately place them in storage.

The STARTGAS and GASPRICE fields are crucial for Ethereum's anti-denial of service model. In order to prevent accidental or hostile infinite loops or other computational wastage in code, each transaction is required to set a limit to how many computational steps of code execution it can use. The fundamental unit of computation is "gas"; usually, a computational step costs 1 gas, but some operations cost higher amounts of gas because they are more computationally expensive, or increase the amount of data that must be stored as part of the state. There is also a fee of 5 gas for every byte in the transaction data. The intent of the fee system is to require an attacker to pay proportionately for every resource that they consume, including computation, bandwidth and storage; hence, any transaction that leads to the network consuming a greater amount of any of these resources must have a gas fee roughly proportional to the increment.

Messages

Contracts have the ability to send "messages" to other contracts. Messages are virtual objects that are never serialized and exist only in the Ethereum execution environment. A message contains:

The sender of the message (implicit)
The recipient of the message
The amount of ether to transfer alongside the message
An optional data field
A STARTGAS value

Essentially, a message is like a transaction, except it is produced by a contract and not an external actor. A message is produced when a contract currently executing code executes the CALL opcode, which produces and executes a message. Like a transaction, a message leads to the recipient account running its code. Thus, contracts can have relationships with other contracts in exactly the same way that external actors can.
Note that the gas allowance assigned by a transaction or contract applies to the total gas consumed by that transaction and all sub-executions. For example, if an external actor A sends a transaction to B with 1000 gas, and B consumes 600 gas before sending a message to C, and the internal execution of C consumes 300 gas before returning, then B can spend another 100 gas before running out of gas.

Applications

In general, there are three types of applications on top of Ethereum. The first category is financial applications, providing users with more powerful ways of managing and entering into contracts using their money. This includes sub-currencies, financial derivatives, hedging contracts, savings wallets, wills, and ultimately even some classes of full-scale employment contracts. The second category is semi-financial applications, where money is involved but there is also a heavy non-monetary side to what is being done; a perfect example is self-enforcing bounties for solutions to computational problems. Finally, there are applications such as online voting and decentralized governance that are not financial at all.

Token Systems

On-blockchain token systems have many applications ranging from sub-currencies representing assets such as USD or gold to company stocks, individual tokens representing smart property, secure unforgeable coupons, and even token systems with no ties to conventional value at all, used as point systems for incentivization. Token systems are surprisingly easy to implement in Ethereum. The key point to understand is that a currency, or token system, fundamentally is a database with one operation: subtract X units from A and give X units to B, with the provision that (1) A had at least X units before the transaction and (2) the transaction is approved by A. All that it takes to implement a token system is to implement this logic into a contract.
The basic code for implementing a token system in Serpent looks as follows:

```
def send(to, value):
    if self.storage[msg.sender] >= value:
        self.storage[msg.sender] = self.storage[msg.sender] - value
        self.storage[to] = self.storage[to] + value
```

This is essentially a literal implementation of the "banking system" state transition function described further above in this document. A few extra lines of code need to be added to provide for the initial step of distributing the currency units in the first place and a few other edge cases, and ideally a function would be added to let other contracts query for the balance of an address. But that's all there is to it. Theoretically, Ethereum-based token systems acting as sub-currencies can potentially include another important feature that on-chain Bitcoin-based meta-currencies lack: the ability to pay transaction fees directly in that currency. The way this would be implemented is that the contract would maintain an ether balance with which it would refund ether used to pay fees to the sender, and it would refill this balance by collecting the internal currency units that it takes in fees and reselling them in a constant running auction. Users would thus need to "activate" their accounts with ether, but once the ether is there it would be reusable because the contract would refund it each time.

Financial derivatives and Stable-Value Currencies

Financial derivatives are the most common application of a "smart contract", and one of the simplest to implement in code. The main challenge in implementing financial contracts is that the majority of them require reference to an external price ticker; for example, a very desirable application is a smart contract that hedges against the volatility of ether (or another cryptocurrency) with respect to the US dollar, but doing this requires the contract to know what the value of ETH/USD is. The simplest way to do this is through a "data feed" contract maintained by a specific party (eg. NASDAQ) designed so that that party has the ability to update the contract as needed, and providing an interface that allows other contracts to send a message to that contract and get back a response that provides the price.
Given that critical ingredient, the hedging contract would look as follows:

> Wait for party A to input 1000 ether.
> Wait for party B to input 1000 ether.
>
> Record the USD value of 1000 ether, calculated by querying the data feed contract, in storage, say this is $x.
> After 30 days, allow A or B to "reactivate" the contract in order to send $x worth of ether (calculated by querying the data feed contract again to get the new price) to A and the rest to B.

Such a contract would have significant potential in crypto-commerce. One of the main problems cited about cryptocurrency is the fact that it's volatile; although many users and merchants may want the security and convenience of dealing with cryptographic assets, they may not wish to face that prospect of losing 23% of the value of their funds in a single day. Up until now, the most commonly proposed solution has been issuer-backed assets; the idea is that an issuer creates a sub-currency in which they have the right to issue and revoke units, and provide one unit of the currency to anyone who provides them (offline) with one unit of a specified underlying asset (eg. gold, USD). The issuer then promises to provide one unit of the underlying asset to anyone who sends back one unit of the crypto-asset. This mechanism allows any non-cryptographic asset to be "uplifted" into a cryptographic asset, provided that the issuer can be trusted.
In practice, however, issuers are not always trustworthy, and in some cases the banking infrastructure is too weak, or too hostile, for such services to exist. Financial derivatives provide an alternative. Here, instead of a single issuer providing the funds to back up an asset, a decentralized market of speculators, betting that the price of a cryptographic reference asset (eg. ETH) will go up, plays that role. Unlike issuers, speculators have no option to default on their side of the bargain because the hedging contract holds their funds in escrow. Note that this approach is not fully decentralized, because a trusted source is still needed to provide the price ticker, although arguably even still this is a massive improvement in terms of reducing infrastructure requirements (unlike being an issuer, issuing a price feed requires no licenses and can likely be categorized as free speech) and reducing the potential for fraud.