Hyperledger

Hyperledger is a collaborative effort from different industry leaders to frame an open source, Cross-Industry Blockchain aided technologies. The movement basically aims to develop the distributed ledgers that can support enterprise-level business transactions. The entire project is developed on the open source platform. Even though the project is hosted and driven by the free folk of the internet 'Linux Foundation', technology giants like IBM, Intel, Samsung and many more others already became part of the project.

The project was announced in December 2015 by Linux foundation, and soon it became popular as leaders from different business domains like banking, healthcare, finance, supply chain, IoT, manufacturing etc. joined the movement. As of now with 170+ members, the project is the largest blockchain technology consortium and it is entirely funded by its members. Linux Foundation does not stipulate a single blockchain standard for the participants, rather they choose a community-driven approach to develop blockchain technologies. By early 2016, the project began accepting proposals for incubation and later a number of different business blockchain frameworks and tools were accepted for incubation under this project.

Under the project following frameworks have been unveiled so far.

- **Iroha**
- **Fabric**
- **Sawtooth**
- **Burrow**
- **Indy**

Another important thing to point out is the difference between Hyperledger and

Digital Tokens

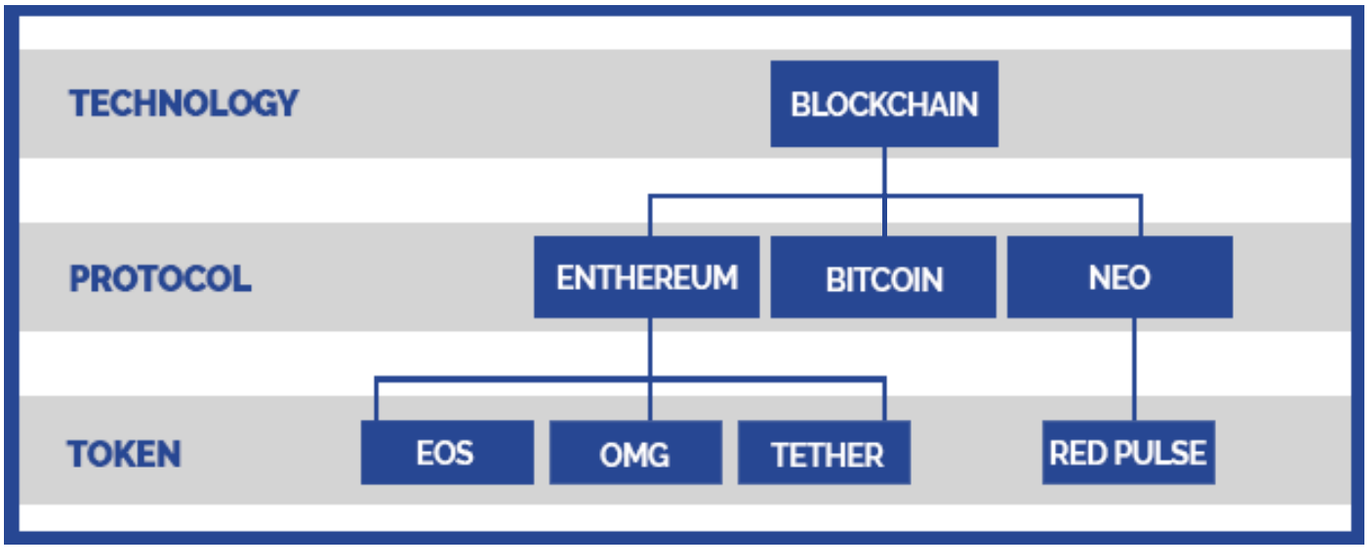
Digital tokens or simply Tokens, are another trending blockchain based application which is shaking the market. So what are they? Tokens are a slight variant of cryptocurrencies; they are a digital asset which is built on top of cryptocurrency of a blockchain network. The token can be used to provide a right, to pay for a service or to transfer data, as an incentive, as a gateway to extra services or any other purposes. In other words, a token can be used in whichever way the developer/ the developing organization decides.

The tokens can be mainly classified into two; Utility tokens and Equity tokens. Utility tokens or user tokens will provide some future access to a product/ service to the user.

Equity tokens are a subcategory of security tokens that represent ownership of an asset, such as debt or company share etc. Equity tokens can be considered as an investment.

The tokens will never be used as a cryptocurrency rather it is a digital asset which is less liquid than cryptocurrency. So whenever a token is created, its value also will be defined. In some cases, the tokens are refundable, that is we can exchange the token with a cryptocurrency. Like the cryptocurrency, the tokens are also managed with wallets.

See the below image to get a better understanding about tokens.



There are different blockchains exist today, like Ethereum and Bitcoin. We can collectively call them as blockchain protocols. Every blockchain based application including cryptocurrency and Tokens will be working on any of these protocols.

How Tokens are created.

Currently, most of the tokens are created on the Ethereum blockchain. Token creation using Ethereum blockchain is a simple process. In Ethereum, token creation is nothing but the creation of a smart contract. A Developer can simply create tokens using the development standards like ERC 20. Presently, there are more than 10000 tokens are available in the ERC20 token standard. The ERC 20 has a predefined structure for smart contract creation. For developing the tokens one just have to add necessary codes to the structure. Once the token is completed, it can be deployed in Ethereum network

ICO (Initial Coin Offering)

ICO (Initial Coin Offering) is similar to IPO (Initial Public Offering) in the case of stock markets. The ICO is generally used for crowdfunding. A company/an entity will offer a new Cryptocurrency/Token to investors against any other cryptocurrency or fiat currency. The investor can keep the token and exchange it in future for a service/product/anything that is assured by the issuer. The return on any token depends upon the smart contract of the token (which is of course defined by the issuer). Like the shares in IPO, the tokens issued through ICO is tradable also. The value of the token changes corresponding to its demand, the token holder can trade the tokens in such situation.

ICO has become a popular way of fundraising for startups, charity, and many other programs. Unlike any other written contract, the smart contracts of a token can’t be modified by the issuer over the time. So ICO offers an assured return for investors if the issuer finds their fortune.

Many companies and startups accumulated massive funding in recent past through ICO. Some examples are,

OmiseGO

OmiseGO (OMG) token was issued by the Omise aimed at raising funds for the creation of a decentralized platform for the exchange of fiat money and crypto-currencies. Initial coin offering helped the project to raise $19 million. Any user of Omise GO will be able to conduct financial transactions such as payments, remittances, payroll deposit, B2B commerce, supply-chain finance, loyalty programs, asset management and trading, and other services, in a decentralized and in-expensive way.

EOS

EOS is token issued by the company ‘Block. one’ to conduct the Proof of Stake mechanism in blockchain development. Basically, EOS token holders vote for the block producers, which mine blocks and decide on major events in the EOS ecosystem.

Tether

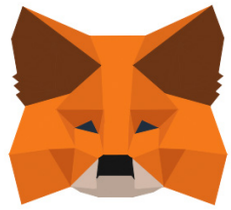
A method to maintain a one-to-one reserve ratio between a cryptocurrency token, called tethers, and its associated real-world asset, fiat currency.

SECTION-2

Cybrosys Limited Edition

BLOCKCHAIN

E-BOOK



MetaMask

MetaMask is a web browser add-on which enables anyone to run the Ethereum DApps without running the Ethereum full node. An Ethereum full node installation will take a lot of memory as well as time; so Metamask is a tool that eliminates the overburden of this hectic installation task. Initially, Metamask was available only for Google Chrome, but now it is available for Firefox and other popular web browsers.

MetaMask add-on for chrome can be added from chrome web store or from 'metamask.io' website. This MetaMask add-on provides a user interface for interacting with the blockchain. The user can connect to the Ethereum main network or 'testnet' or he may create his own private network and run DApps on the blockchain.

In normal case, a web3.js (the JavaScript API for Ethereum DApps) must be installed in the local system to interact with the Ethereum DApps. Web3.js is a collection of libraries used to interact with local or remote Ethereum node using Http or IPC connection. But MetaMask will inject the web3.js to each page for accessing the Ethereum blockchain by itself. This approach eliminates the effort of web3.js installation in the local system.

After adding the Metamask, the user can interact with Ethereum blockchain as normal. The user can create an account, access Ethereum DApps, or deploy once own DApp. MetaMask retrieves data from the blockchain and allows the users to manage the data securely.

The Metamask provides a vault account for each user, this vault secures, stores and tightly controls access to tokens, password, certificates, API keys and other

elements in blockchain apps. The vault account act as a second level encryption for the user account.

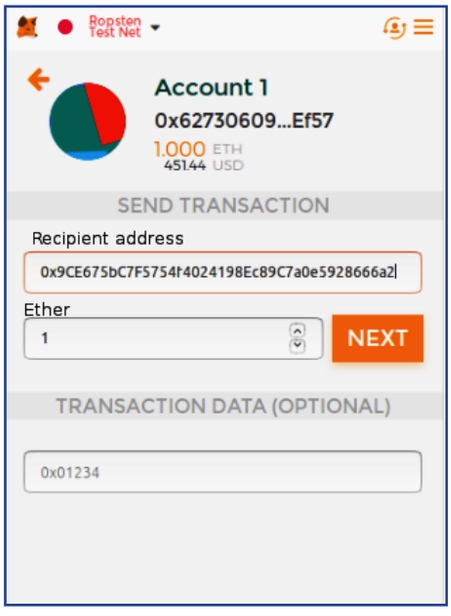
Wallet Seed

The Metamask will provide a group of 12 words known as “wallet seed” while installing it. It is the user credential and it must be stored somewhere safe. The users can also create passwords for their account. The wallet seed or the password is necessary to log in to the MetaMask. The vault account will encrypt the user metadata and securely store it in the browser itself.

MetaMask Transactions.

The Metamask user interface has a default buy and send option for buying and sending Ether. The user can access his wallet, buy or send ether, check his balance and transactions from this interface. When the user executes a transaction from the Metamask it will send the transaction to the respective blockchain network. Then the corresponding validation and confirmation will occur in the blockchain as usual. In the case of ‘testnet’ and main network, the user can see the transaction details and confirmations in the ‘Etherscan.io’.

Here is an example of a transaction of Ether through Metamask.
For sending Ether to an account you have to specify the recipient address and the amount to be transferred in the provided interface.



Before linking your transactions to the blockchain, the web3.js will ask your permission and the transaction will be submitted only after your approval.

Once the user submits the transaction, it will be sent to the blockchain for validation. The transactions will be broadcasted to the nodes and once the validation is completed you can see the transaction details in the etherscan window. The window will display all the information regarding that transaction i.e.; block number, hash value, sender and recipient address, number of confirmations, gas units, transaction cost, nonce etc.

GNOSIS, Maker(MKR), Token Factory, CryptoKitties etc. are some DApps that supports Metamask. Any developer can submit DApp in Ethereum with Metamask support so that the user doesn't need to install the full Ethereum node for accessing the particular app. The Metamask is the very useful tool for accessing Ethereum in low bandwidth networks. Let's hope the tool will expand the reach of 'Ethereum' to more people.

Fabric. Often people are confused with these two terminologies. The fabric is one of the protocols under the hyperledger project and Hyperledger is the name of the project itself, not a technology.

So if we recall the concepts discussed so far, Bitcoin is a blockchain network which offers only one kind of application that is bitcoin itself. Ethereum is a more versatile blockchain platform where anyone can develop and deploy the blockchain based application. Hyperledger is a project which aims to develop different enterprise level distributed ledger technologies.

The tools that are currently under hyperledger project

- Hyperledger cello
- Hyperledger composer
- Hyperledger explorer
- Hyperledger quilt

The project along with its participating technologies are often referred as 'Hyperledger Umbrella'

Objectives of the Hyperledger Project.

- To enable its member organizations to build robust, enterprise-level applications, platforms and hardware systems based on blockchain technology
- To advance the use of blockchain technology in business by developing a cross-industry level, open-source development library
- To facilitate building custom distributed ledger solutions for its members.
- To integrate independent blockchain protocols.

Mist

Mist is an Electron framework based application which is used for the management of Ethereum wallet and Ethereum applications. More specifically, it is a hybrid desktop application with a web interface to manage Ethereum DApps (Decentralized Apps). Mist is similar to a web browser like Chrome or Firefox, but a Web 3.0 edition. Mist is an all in one software to manage all the assets and contracts of an individual in the Ethereum Blockchain. Mist can browse DApps, manage contracts, manage ether and other digital assets etc. It acts like a window to the Blockchain network and access different Apps and Services provided in the network. Mist is also the official wallet of Ethereum blockchain to manage 'Ether'. Mist is developed and maintained by Ethereum team and it is still in beta stage, hence it may have problems. It is recommended to use the latest and updated version of the official Git repository of Mist.

Mist installation is similar to any other software. Download the 'Mist' executable file appropriate to your operating system from Git repository and install it like any other application. Since Mist is a client based application, it has to download and save the entire Ethereum Blockchain to the local system. As of now, there is at least 5GB of data and it will increase as the chain grows. So the downloading and installation process is a big task.

Mist wallet

Mist wallet is the official wallet of Ethereum Blockchain so the Ethereum currency Ether can be stored and managed with it. Since Mist can serve both as a wallet and asset management tool it is easy to engage in Ethereum transactions using mist. Another advantage of using Mist is the security. The wallet is purely designed and developed by Ethereum itself not by any third party. Mist wallet provides two type of service namely 'Simple Wallet' and 'Multisignature wallet'. In Both services, the users can manage their funds, ethers, as well as the contract in simple steps. But the 'Multisignature wallet' can provide some

extra layer security to its user.

An important thing to note while using a Mist wallet is about syncing process. Always ensure that the entire Blockchain is synced with the local system. Incomplete syncing may invite critical problems during the transaction.

Truffle

Truffle is a blockchain development framework dedicated to Ethereum blockchain platform. This open source framework was developed by Consensys with an objective to simplify the blockchain and DApp development in Ethereum platform. Truffle is a well-equipped framework which makes Ethereum platform more user-friendly and interactive. Following features of truffle make it the best choice for developments in Ethereum.

Features of Truffle

1) Automated tools

- Truffle provides built-in smart contract creation tools, which will perform compilation, linking, deployment and binary management of the smart contracts. It offers a development environment with a testing framework and digital asset pipeline. The truffle reduces the complexities of team-based development, testing, deployment, and migration activities in an Ethereum project. In total, smart contract creation and testing which are two tedious tasks in blockchain development have become easier with truffle.

2) Scriptable

- Along with those automated tools available, the truffle is scriptable too. Which means the developer is allowed to add more scripts to the project during the development. For running these scripts a 'script runner' is also available in truffle, which can run both external and internal scripts.

3) Networking

- As the blockchain is growing day by day extensions and migrations have

become a frequent requirement in the development environment. The truffle framework is fully capable of integrating an infinite number of networks to it and will completely support migrations and extensions of networks. The configurable build pipeline in truffle supports tight integration within the network.

4) Package support

- Truffle provides two package management tools, Ethpm & Npm. Ethpm is the Ethereum Package Manager and Npm is Node Package Manager These package includes several modules and set of predefined codes which can be utilized while scripting.

5) User interaction

- Truffle has an interactive console for the direct contract communication. Through this console, the developer can create, compile and test the smart contracts. The console also manages the communications between the user and smart contracts.

Another interaction environment available in truffle is the browser portal. The developer can use the default account in truffle to use the browser environment. It is possible to see the local transactions, pending requests etc. from there.

Development-Truffle boxes

Before installing a truffle environment owe must install an Ethereum client like 'Geth', 'WebThree', Parity' or any other. Truffle is installed as a separate project above Ethereum client and it can be deployed on it without any extra configuration.

After installing the Ethereum client, create a project directory for truffle using the command

mkdir projectfolder'

To initialize the project (i.e. truffle) navigate to the created directory

cd projectfolder
And run following command.

init truffle
If the initialization is complete then the following project structure will be created

1. **Contracts**
2. **Migrations**
3. **test**
4. **truffle.js**

By default, there will be some sample contracts and set of sample projects in truffle environment to get acquainted with the truffle environment.

Truffle Box

In truffle, each individual project is called truffle boxes. The truffle box will contain required modules, front end views, solidity smart contract libraries of a

project etc..

As mentioned earlier, Truffle comes with some sample Truffle Boxes available. The user can unbox and run them in the test environment.

Creating a Truffle Box

Users can create their own truffle boxes in the truffle environment. They can either create truffle from scratch or can add an existing project to the box and develop from it. If the project is starting from scratch a 'blueprint truffle box' is readily available in the truffle. The 'blueprint truffle box' will contain all the necessary configuration files and common values for a truffle box.

For developing truffle boxes the user need a

1. **Github repository,**
2. **Configuration file,**
3. **Optionally small and large images for the boxes listing.**

The truffle configuration file name is *truffle-box.json*. It contains 3 attributes ignore, commands and hooks

Ignore: An array attribute contains the list of files to be ignored while unboxing.
Commands: Object attributes of a key-value pair, contains the list of files to be compiled or migrated. After unboxing these files are visible to users.
Hooks: Object attribute which contains a list of commands need to be executed when unboxing.
The blueprint contains all the basic components needed for developing a truffle box. The developer can delete the default sample files and images from the box and can create new files. The configuration file is also customizable.

Community truffle box

Truffle is already providing several official truffle boxes developed by the truffle developers. Along with that a vibrant truffle community is also actively contributing new truffle boxes. Any individual can contribute to the community by sending the developed truffle details and GitHub repository details to truffle community. The Box will undergo a screening and if it is compatible with truffle then it will be published as a community truffle box

Embark

Embark is a framework used for developing and deploying DApps (Decentralized Apps) using one or more decentralized technologies. The tools and functionalities provided by the Embark make the DApp development process easy and productive. It reduces the interaction between the front end of application and smart contract so that the application can run faster on the network. The technology uses IPFS protocol to store and manage files across the decentralized network. 'Whisper' and 'Orbit' communication platforms streamline the communication process in Embark. The platform performs automatic deployment of smart contract and ensures the redeployment if the contract has undergone any changes. Embark currently integrates with Ethereum, decentralized storages like IPFS, and decentralized communication platforms like Whisper and Orbit. Using embark, it is possible to manage different chains like testnet, private net, livenet etc. And the smart contracts in solidity and serpent can be built and it is deployable with embark.

How to Install embark?

Prerequisites:

- geth (1.5.8 or higher)
- node (6.9.1 or higher)

Open terminal and run the below-given codes

```
npm -g install embark
```

To run it on a simulator instead of a real Ethereum node, run the code

```
npm -g install ethereumjs-testrpc
```

Creating a DApp with Embark

Generally, a typical DApp in embark consists of 2 sections.

1. Smart contracts

this section will contain the business logic of the DApp. It is written in either solidity or serpent

2. User Interface

the user interface through which the user interacts with the app

To create a DApp in embark simply run this code

embark new <DApp name>

This code will initialize a predefined template and creates a directory structure