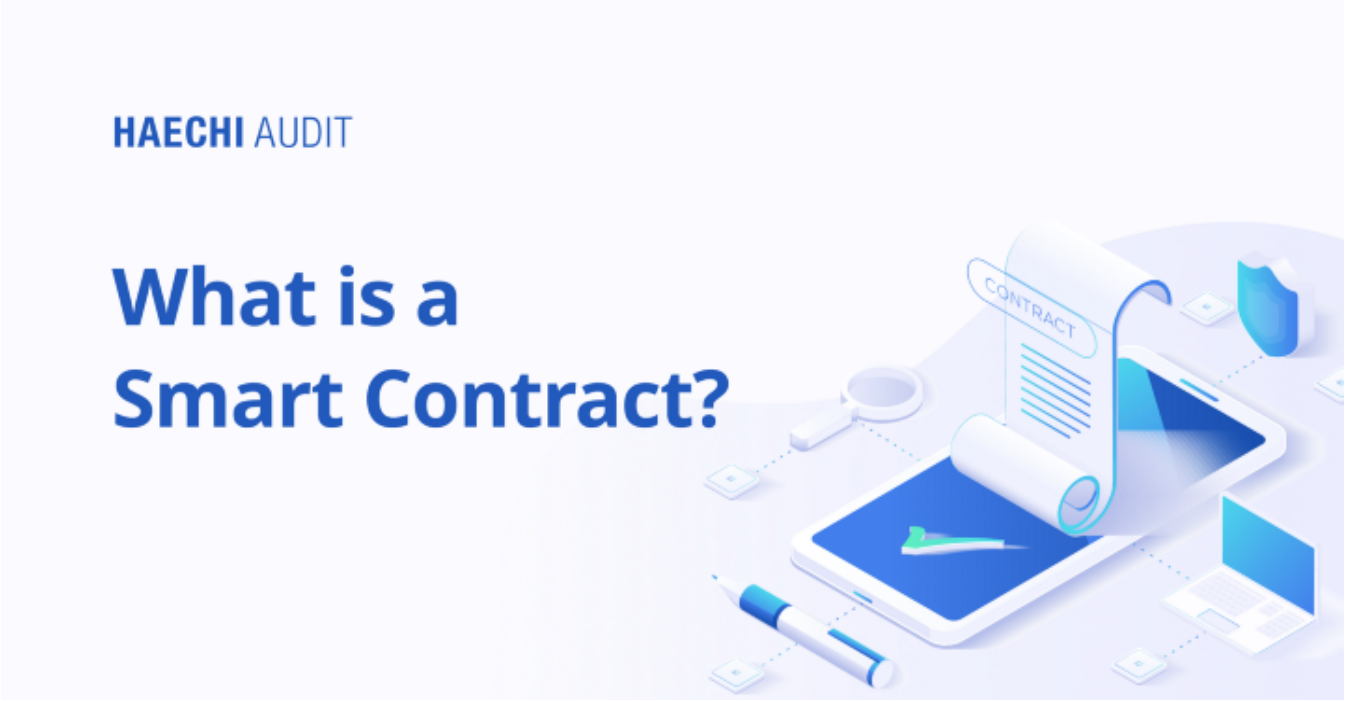
Published in HAECHI AUDIT

Se Jin OH (Laura) Follow
Jan 6, 2020 · 10 min read · Listen

What is a Smart Contract? A beginner's guide to Blockchain

Introduction to Ethereum smart contracts and how they work

Co-authored by @laura_oh @nipolnipol



'Don't smart contracts work for everything?'

In a recent gathering, the topic of blockchain came up without exception. People were especially interested in actual applications of blockchain technology, and one of the participants told us she was researching ways to improve the tagging system for pets using blockchain. When I asked how they were planning to apply

On my way home, I realized the necessity of writing an article explaining smart contracts. This article explains what smart contracts are, how they function and how they can be used.🤓

Smart Contracts Are...

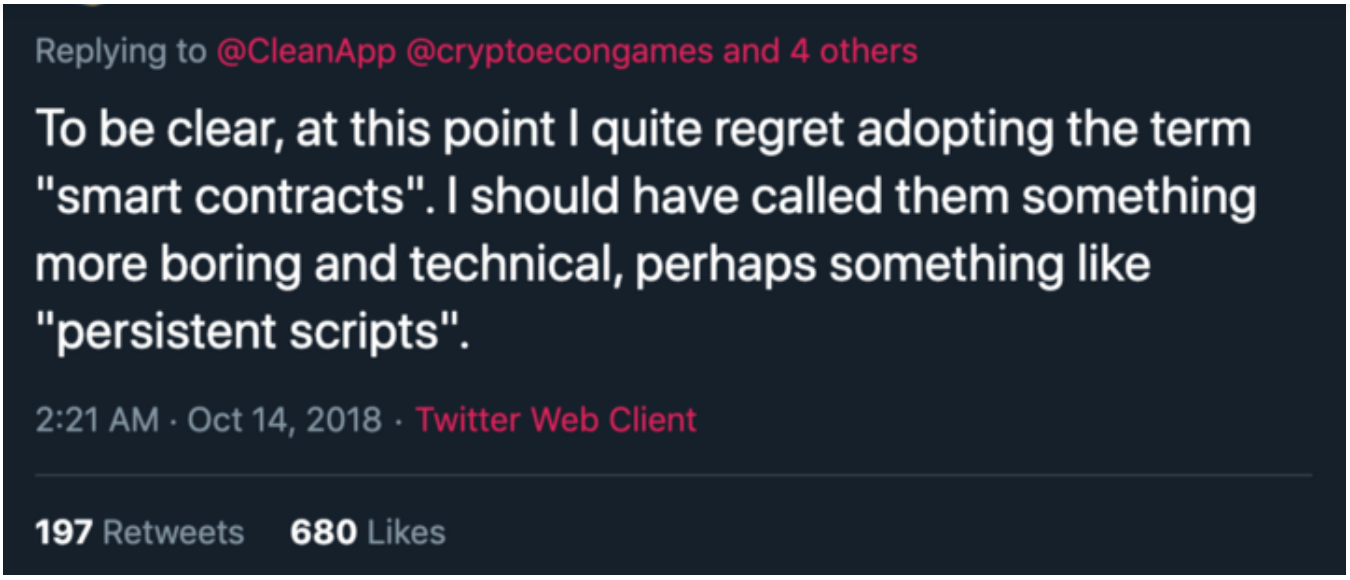
Scripts that implement paper contracts with code lines, and ensure the contracts are put into effect when certain conditions are met

- **Background**

Some people think the concept of smart contracts appeared along with blockchain, but this is hardly true. The idea of a smart contract was suggested in 1994, by computer science/law major Nick Szabo.

In his paper 'Formalizing and Securing Relationships on Public Networks,' he argued that smart contracts facilitate every process of a contract using protocols and user interfaces. This provides a new method of digital contracting with much more features than the conventional pen & paper contracts, and that smart contracts can reduce transaction costs.

And in 2013, Vitalik Buterin, the co-founder of Ethereum uploaded the whitepaper 'Ethereum: The Ultimate Smart Contract and Decentralized Application Platform' to his blog. Then and there, Vitalik introduced and implemented the concept of smart contracts into the Ethereum blockchain system, opening the widespread use of smart contracts.

It is interesting to note that last year, Buterin tweeted that he regrets the use of the term ‘smart contract,’ and that he should have instead called them something like ‘persistent scripts.’ 😂 This opinion seems to stem from the possibility that the term ‘smart contract’ may be applied too broadly.

- **The concept of blockchain smart contracts.**

Current smart contracts used in blockchain resemble scripts that implement conventional pen & paper contracts with code lines, and execute the contract when certain conditions are met.

Using smart contracts allows that contracts be made between two parties completely unknown to each other; this is because the contract does not take effect if certain conditions are met. It follows that one of the main characteristics of smart contracts is that it facilitates the signing and execution of contracts without middlemen.

Smart contracts are implemented on many blockchain platforms, not limited to Ethereum. Blockchains have numerous witnesses, and draw its results from their multiple inputs, and it was the ideal environment to test the idea of smart contracts.

How Smart Contracts Function

Most smart contracts are built on programming languages. They can be designed to

Unfortunately, Ethereum smart contracts are not compatible with conventional programming languages. Instead, smart-contract specific languages such as **Solidity and Vyper** can be used to define a logical contract function. A compiler is then used to convert this logical contract function into byte codes and distribute them onto blockchains.

More specifically, Ethereum smart contracts function in the following manner :

- What you want to implement with Smart Contracts will be implemented by Solidity.
- Compile the Solidity code and generate Bytecode to distribute over the network.
- The transaction incorporates the bytecode, and the miner mines the block with the transaction. Simultaneously, the transaction is recorded on the blockchain network.
- The user generates a bytecode to call the function defined by the distributed smart contract code using ABI, and deliver it to the blockchain network as a transaction.
- The miner runs the bytecode according to the distributed smart contract code on the EVM(Ethereum Virtual Machine). At this point, Gas fees are calculated and added to the block, and if the executed results are valid, the results are reflected in the blockchain's state.

When a logical contract function is converted into bytecode, it also transfers something called the Application Binary Interface(ABI). An interested party who intends to use the contract does not interact directly with the computer. Since they interact using in/output devices such as keyboards, mice and monitors, they must be constructed in human-friendly interfaces. Using this ABI, the process of constructing this human-friendly interface becomes easier.

To interact with smart contract, the Web3.js library is provided, which helps smart contract functions to be used on web browsers. If one wishes to use these functions in Python, there is also Web3.py, and for mobile environments, one may utilize

smart contract is a simple auction contract available on the Solidity official document. We see it is very similar to the classes we've seen in OOP programming languages such as JAVA.

```
pragma solidity >=0.4.22 <0.7.0;

contract SimpleAuction {
    address payable public beneficiary;
    uint public auctionEndTime;

    address public highestBidder;
    uint public highestBid;

    mapping(address => uint) pendingReturns;

    bool ended;

    event HighestBidIncreased(address bidder, uint amount);
    event AuctionEnded(address winner, uint amount);

    constructor(
        uint _biddingTime,
        address payable _beneficiary
    ) public {
        beneficiary = _beneficiary;
        auctionEndTime = now + _biddingTime;
    }

    function bid() public payable {
        require(
            now <= auctionEndTime,
            "Auction already ended."
        );

        require(
            msg.value > highestBid,
            "There already is a higher bid."
        );

        if (highestBid != 0) {
            pendingReturns[highestBidder] += highestBid;
```

EthFiddle Compile and Run ►

1) Constructor

Function: It sets the bidding time and beneficiary.

```
19          address payable _beneficiary
20      ) public {
21          beneficiary = _beneficiary;
22          auctionEndTime = now + _biddingTime;
23      }
```

We'll take a look at the code on the block level. Lines 17~23 define the constructor. The parameter _biddingTime sets the length of time during which the auction will be open, and _beneficiary sets the beneficiary once the auction is over.

2) bid()

Function: Participate in the auction using ETH.

```
25      function bid() public payable {
26          require(
27              now <= auctionEndTime,
28              "Auction already ended."
29          );
30
31          require(
32              msg.value > highestBid,
33              "There already is a higher bid."
34          );
35
36          if (highestBid != 0) {
37              pendingReturns[highestBidder] += highestBid;
38          }
39          highestBidder = msg.sender;
40          highestBid = msg.value;
41          emit HighestBidIncreased(msg.sender, msg.value);
42      }
```

Now, to participate in the auction, we must run the bid function and send the Ether at a same time. Lines 25~42 contain a check if the current time is within the auction

characteristics are entirely identical, and the database does not reflect changes until the transaction is fully executed. If any of the queries of the transaction fail, the entire transaction fails and any intermediate steps are reverted.

As we can see from the internal logic statements, if the require() statement is unfulfilled, the entire transaction is considered a failure!😲 If the transaction is successful, the previously highest bidder will be saved to the map datatype pendingReturns, and the highest bid will be set to the person who have sent the transaction.

3) auctionEnd()

Function: Ends auction after time is over and delivers ETH to beneficiary.

```
57    function auctionEnd() public {
58        require(now >= auctionEndTime, "Auction not yet ended.");
59        require(!ended, "auctionEnd has already been called.");
60
61        ended = true;
62        emit AuctionEnded(highestBidder, highestBid);
63
64        beneficiary.transfer(highestBid);
65    }
```

Let's say nobody enters additional bids after this time. Then someone will call the auctionEnd() function defined by lines 57~65. The predefined beneficiary receives the highest Ether deposit, and the contract will be terminated.

4) withdraw()

Function: Non-highest bidders can withdraw their ETH.


```
46          if (amount > 0) {
47              pendingReturns[msg.sender] = 0;
48
49              if (!msg.sender.send(amount)) {
50                  pendingReturns[msg.sender] = amount;
51                  return false;
52              }
53          }
54          return true;
55      }
```

Afterwards, non-highest bidders will call the withdraw() function defined by lines 44~55. There is no separate time check, so as long as the contract is valid, withdrawals can be made anytime.

```
emit HighestBidIncreased(msg.sender, msg.value);
```

Have you noticed this line in each function? This is called an event; its purpose is to leave a log of each function and that it was properly executed. It returns a receipt as a result of a transaction, which contains the said event. If the transaction had failed, there would be no event recorded.

If you'd like to look at an upgraded version of this smart contract, you may look at https://solidity.readthedocs.io/en/latest/solidity-by-example.html#id2. 😉

Why Do We Need Smart Contracts?

Before we look at why we need these smart contracts, we'll briefly summarize the characteristics of smart contracts. **From a technological perspective, smart contracts have the following features.**

1) Anyone can distribute the contracts.
2) Anyone can verify the contracts — not just the owner.
3) Execution of the code can be automatized.

and where transaction costs are incurred. Using Etherscan or Decentralized Metadata and Source Code Repository, anyone can look at all contracts that have been distributed.

Additionally, smart contracts are effective tools in setting up trustless networks on the blockchain. If Alice needs to send $50 on the last day of every month, this contract is executable without going through banks or other certified third parties.

Where Can We Use Smart Contracts?

The advent of Ethereum has resulted in smart contracts being used in a variety of manners.

- **Initial Coin Offering (ICO)**

An exemplary case is ICO, where smart contracts were used to create tokens and perform their Initial Coin Offerings. Most of these tokens follow an ERC-20 protocol, which was suggested from the EIP(Ethereum Improvement Proposal) 20, and defines the standards for a token-type smart contract. This EIP defines ERC-20 tokens' standards (for qualities such as token transfers, exchanges, and authorizations) they must follow. On a side note, ERC stands for Ethereum Request for Comment.

**EIP(Ethereum Improvement Proposal) Ethereum is a decentralized network, meaning anyone can contribute with new ideas for improving the Ethereum network through EIP.*

The reason token sales were conducted using smart contracts is that the information contained by the smart contracts is displayed on the blockchain network with complete transparency, ensuring its credibility. Airbloc's director Won-Kyung Ryu, having conducted an actual token sales, commented as below in our interview:

"I think using smart contracts greatly enhances the credibility of token sales to participants because we can display the progress of the sale, the number of tokens issued and sold — all in a transparent manner."


eliminates the need for a separate trading platform, and items can be traded directly within the trading system at desired prices. In this case, each smart contract must be mapped to a specific item, and it becomes important that the contract be discrete and not divisible(as is the case where 1 ETH can be divided into 0.5ETH, etc.). Standards that take this need into account are ERC-721 and ERC-1155. For items designed with this protocol, the ownership lies with the individual instead of the developer, and can be traded with or without the developer's approval. Also, this characteristic would be useful in proving ownership of a financial product by utilizing these NFTs(Non-Fungible Tokens).

In addition to fundraising and game items, smart contracts are being used in various sectors such as decentralized finance(De-fi), wallets, decentralized exchange institutions, etc.

The Limits of Smart Contracts

Even so, smart contracts do not perfectly solve the limits of conventional contracts.

Smart contracts cannot retrieve information outside the blockchain by themselves. In other words, we might encounter problems when certain information is necessary during the process of verifying whether smart contract conditions are met. For example, let's say we have a contract that says 'If the price of ETH rises above $1,000, A provides B with 1 ETH.' Here, the smart contract only functions if information on whether the price of Ethereum is over $1,000 or not. Therefore, the contract is only executable when the cryptocurrency exchange's accurate Ethereum price data is retrieved, which lies outside the blockchain.

Here's the catch — during the retrieval process, the wrong data may be retrieved, or the data may be intentionally altered. In other words, specific information about a specific condition needs to be retrieved, and if this information is dependent on a third party, this inevitably leads to a credibility issue. To resolve this inherent issue, many startups are striving to provide accurate information necessary for generating and executing smart contracts.

Another inherent limitation of smart contracts is that they are not editable once they have been distributed. On one hand, it is this rigidity that imbues smart


most prominent case being SmartMesh. The smart contract distributed by the SmartMesh team contained a security flaw, and the hacker used it to issue/sell additional tokens. Also, there was a case where the security flaw resulted in 513.774.16ETH contained in wallets became permanently non-withdrawable.

Smart contracts deal directly in digital assets, and therefore safety is paramount when developing smart contracts. In the blockchain industry, it is becoming standard procedure to conduct a security audit before distribution. We will look at how security audits are performed in further detail in future articles.

Additional Comments

This article was to look at smart contracts as an effective method to solve many of the inherent limitations of the conventional contracting system, and that they are being used in various fields. We've also noted that smart contracts are not able to solve *every* problem with the contract system.

What's interesting about smart contracts is that it is continuously improving as the developing environments centered around the blockchain developers' community also improves. We hope this article serves those who have been interested in blockchain and smart contracts, and any questions and/or feedback is always welcome at audit@haechi.io.

Thank you for reading. 🙆

HAECHI AUDIT Official website: https://audit.haechi.io/

HAECHI AUDIT Twitter: https://twitter.com/haechi_audit

[About Us]

HAECHI AUDIT is a leading entity in the global blockchain industry specializing in smart contract security audits and developing. We are made up of experts with multiple years of experience in blockchain technology research and development, and provide the most reliable smart contract security auditing and developing services.

Our hallmark portfolio includes SK Telecom, Kakao's blockchain subsidiary Ground X, Carry Protocol, etc.; we also pride ourselves in having conducted security audits for over 50 global projects.

Also, based on our technological expertise, we received/ are receiving support from multiple sources such as Samsung Electronics, the City of Seoul, KB Financial Group, Shinhan Bank, Hanwha Group, etc; we have also been awarded subsidies from the Ethereum foundation.

- Contact : audit@haechi.io
- Official : https://audit.haechi.io/