

Understanding Smart Contracts: The Backbone of DApps on Blockchain

As blockchain technology continues to find new ways to seep into different aspects of our lives and endeavours, one thing is certain: this disruptive technology is here to stay. The blockchain opens new opportunities, providing better, more secure, and more efficient alternatives to centralized methods in almost every sector, and smart contracts are a huge part of what makes that possible. In this article, we examine what they are, how they work, as well as how they form the backbone of one of the blockchain's primary features – decentralized apps, which make blockchain disruption across various sectors possible.

First, what are smart contracts?

In the real world, contracts exist everywhere. From real estate deeds to letters of employment, they act as legally binding documents that state the terms of an agreement between two or more parties. In the blockchain, however, smart contracts, mostly running on the Ethereum Blockchain, act as a digital version of such agreements. They do not contain any legal terms in themselves, but instead; they are written as computer programs (code) stored on the blockchain and are set to run or execute automatically as soon as the conditions stipulated therein are fulfilled, using if/then conditional statements within the code itself.

Due to their nature, smart contracts offer the following features and benefits:

1. **Automation:** A smart contract is automated, running according to the codes that power it. Once that code's specifications are met, the contract is immediately executed.
2. **Speed:** With smart contracts, there is no need for the bother of paperwork, or the delay usually associated with centralized institutions. A smart contract is executed once the conditions specified in its programming are met.
3. **No need for intermediaries:** Smart contracts are built upon the decentralized nature of the blockchain. Not only is no central authority needed to verify the terms and transactions that might occur in a smart contract, but it is also self-executing, which means that once the required conditions are fulfilled, no intermediary needs to enforce contract terms. It happens automatically.
4. **Accuracy/No Grey Areas:** Smart contracts are written in code, which means that there is no room for grey areas as can be seen with written contracts, which can twist and manipulate the intended meaning of language, based on how they are worded and written.

5. **Immutability and Security:** Finally, because smart contracts are run on the blockchain, this means they also follow the same rules as the blockchain. So as with all other blockchain transactions, once a smart contract is executed and added to the existing operations/transactions, it cannot be reversed. This helps to reduce the risk of fraud as any attempt to modify the transaction after it has been added to the blockchain is incredibly difficult and would be counter-intuitive to the hacker because of the computation power that would be expended in an attempt to alter the blockchain.

How Smart Contracts Work

Smart contracts work as a series of programmed digital statements (called conditional if/then statements) that are created inside a blockchain. Bitcoin was the first network to support basic smart contracts, but it was highly restrictive and only existed in a way that enabled a transfer of financial value (Bitcoin, the cryptocurrency) from one person to another. The nodes that made up the network would only validate and approve the transaction if all its requirements were met. However, Ethereum decided to improve on this and decided to allow developers to write their own programs, the same way Android and iOS allowed developers to create apps on their platforms. In Ethereum's case, it allowed developers to code their own smart contracts in a language called Solidity, its programming language for smart contracts.

Creating smart contracts follows a simple process, which can be outlined as follows:

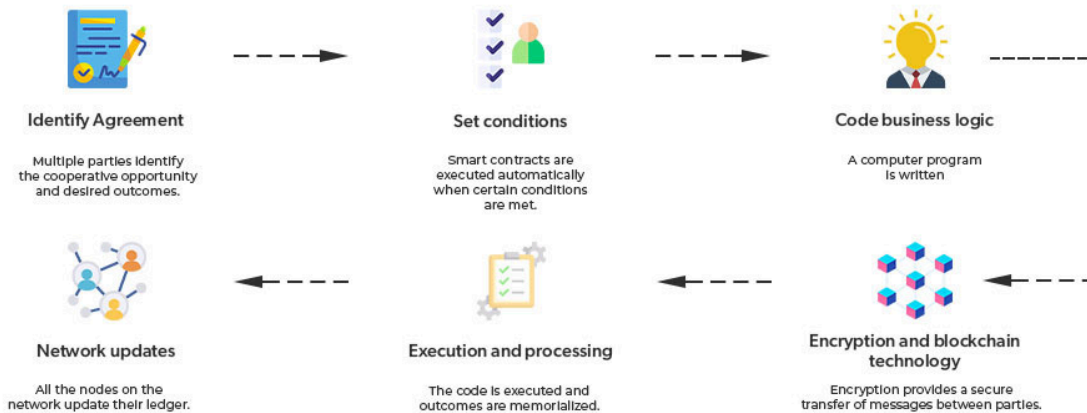
First, the parties involved come to an agreement on what the contract's terms should be. Here, the obligations of both parties, other important information, as well as the conditions for contract execution are laid out and well-defined.

Second, developers use Solidity (if they are using the Ethereum blockchain) or other programming languages such as to create the script for the smart contract. This script is tested and then checked for any issues or vulnerabilities by a team that deals in smart contract audits, such as us here at Credulen. This is necessary because once deployed on the blockchain it becomes immutable and cannot be undone; and deploying a flawed contract can have disastrous effects, such as the MakerDAO incident.

The third step is to deploy the already tested smart contract on the blockchain platform of choice (such as the Ethereum, Solana or Polygon networks). Here, the smart contract 'goes live', so to speak. Once this is done, the smart contract is then linked to one or several 'oracles', third-party services that allow smart contracts to connect to real-world data sources that are not on the blockchain, such as flight details in our insurance example below. This enables the smart contract to 'be aware' of conditions related to external sources so that once they are met, the contract is executed.

Finally, once the circumstances and conditions are satisfied, the contract is automatically executed. Upon execution, the contract is entered into the ledger.

How does a Smart Contract Work?



How Smart Contracts Work - A simplified process. Image: [GeeksforGeeks](https://www.geeksforgeeks.org/smart-contracts/)

Based on this process, smart contracts [find application in various fields](#) including real estate, insurance and even art.

Here are a few if-then models in various fields:

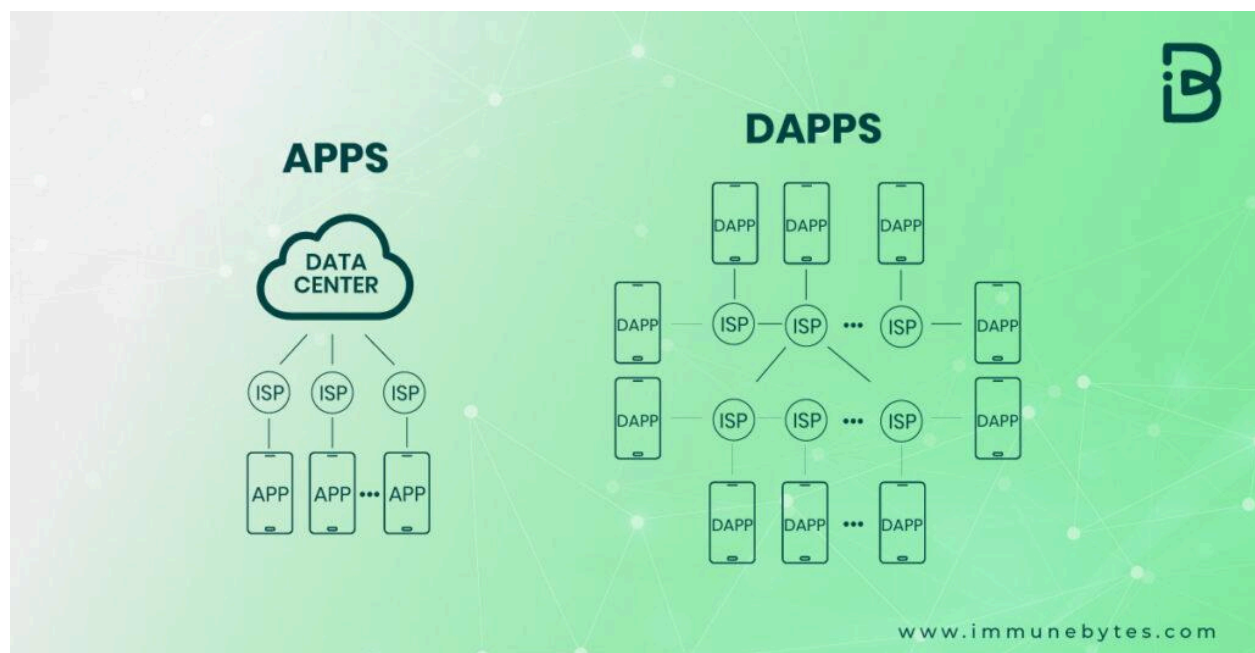
- **Agriculture**: Let's say a market asks a farmer for 100 ears of corn. The former will lock funds into a smart contract that can then be approved when the latter delivers. When the farmer delivers their obligation, the funds will immediately be released — i.e., after the fulfilment of a legal contract. However, the contract is cancelled and funds are reversed to the client if the farmer misses their deadline. (Source: [CoinTelegraph](#))
- **Insurance**: Rachel is at the airport, and her flight is delayed. AXA, an insurance company, provides flight delay insurance utilizing Ethereum smart contracts. This insurance compensates Rachel in such a case. How? The smart contract is linked to the database recording flight status. The smart contract is created based on terms and conditions.

The condition set for the insurance policy is a delay of two hours or more. Based on the code, the smart contract holds AXA's money until that certain condition is met. The smart contract is submitted to the nodes on EMV (a runtime compiler to execute the smart contract code) for evaluation. All the nodes on the network executing the code must come to the same result. That result is recorded on the distributed ledger. If the flight is delayed in excess of two hours, the smart contract self-executes, and Rachel is compensated. Smart contracts are immutable; no one may alter the agreement. (Source: [Simplilearn](#))

Smart Contracts and decentralized applications (DApps)

Smart contracts also function as a 'backbone', or powerhouse of decentralized apps (otherwise known as DApps), blockchain-based applications – mostly hosted on the Ethereum Network - that have now become a very important part of the blockchain ecosystem.

Think of it this way. Just like mobile apps allow us to access various features and do many things on our smartphones or laptops that would not normally be possible on those devices, decentralized apps also allow users to access a wide variety of functions that would not normally – in themselves - be possible on the blockchain without them. They use the power of smart contracts, which allow users to create various use-cases or functions across different fields, and a user interface, usually available on the web, as mobile applications, or on desktops and laptops. However, unlike normal mobile or desktop apps which store data on centralized servers, DApps are decentralized, as they operate on the blockchain. No single individual or corporate entity holds your data; instead, it is stored through all the distributed nodes on the blockchain.



In DApps, everyone owns everything here. Image via [Immunebytes](https://www.immunebytes.com)

Because decentralized apps operate on smart contracts, they can apply to almost any industry ranging from finance (which is where DeFi or decentralized finance comes in) to entertainment to even medicine; provided the smart contracts that power them are configured correctly. With the right set of smart contracts, there is almost no limit to where decentralized apps can operate.

A financial DApp would mean no delays with financial transactions as funds are transferred between users in the blockchain, across borders, without a central bank to delay the process. A social media DApp, one wouldn't need to worry about your data being collected and sold for profit because no data is held by anyone. There's also no physical server that can be targeted in

a hack. A voting DApp can make voting incredibly near-stressless and easy, allowing users to vote anonymously under a predetermined set of rules. Smart contracts cannot be rigged in favour of one person or another, they only answer to the conditions set in the code; with the results being stored on the blockchain, making them tamper-proof. From simple functions like enabling automatic transactions to the distribution of digital art, the use cases are endless.

Challenges of Smart Contracts

Smart contracts aren't without their limitations, however. Even though they possess a wide variety of applications; and are generally secure and tamper-proof; several factors still pose a challenge to the large-scale usability of smart contracts. These include:

1. Dependency on external data

A major limitation of smart contracts is that they are not connected to real-time data. They rely on third-party data sources, called oracles (mentioned earlier), which function in the same way the Internet does with computers; connecting the smart contracts to retrieve information from the outside world. While these oracles expand the capabilities of smart contracts; they can also have vulnerabilities or inaccuracies because they are susceptible to manipulation and tampering.

2. Code vulnerabilities

Just like any software, smart contract codes can contain errors, bugs, and other vulnerabilities prior to deployment that can be exploited by hackers. Errors in the code's design can lead to smart contract hacks that could result in both financial loss and reputation damage, such as with [The DAO in 2016](#), or [the Poly Network in 2022](#). Because of this, it is well agreed that smart contracts need to pass through levels of security audits and rigorous testing before being launched to correct these errors and reduce the risk of exposing users to them when the smart contract goes live.

3. Immutability

Another limitation is a feature that is actually one of its benefits: immutability. Since a smart contract becomes immutable (ie. cannot be altered) once it is executed on a blockchain, this means that faulty code - if noticed late, cannot be corrected, and most smart contracts cannot be updated if the circumstances under which it was written become different and if things need to be altered, a new version of the contract has to be created, as well as a way to link the old version to the new one. This can be done using proxy patterns, which helps to retain a smart contract's immutability while being able to update it even after deployment.

Conclusion

Smart contracts are a revolutionary part of blockchain infrastructure, allowing for multiple possibilities and functionalities across various fields and industries. Their speed, immutability, as well as their ability to be self-executing which ensures trustlessness (parties not needing to trust each other) as well as eliminates the need for intermediaries, all contribute to why they have become an important feature and a backbone behind decentralized applications; creating new layers of opportunity.