Blockchain Underpinnings: Hashing - ConsenSys Media

Cryptographic hash functions - Wikipedia

Cryptographic hash functions - Kahn Academy

Merkling in Ethereum by Vitalik Buterin

Ever Wonder How Merkle Trees Work - ConsenSys Media

Patricia Merkle Trees - Ethereum Wiki

Blockchain Demo videos

Bitcoin Wiki - Blockchain

Fat Protocols by Joel Monegro

Solidity Documentation Blockchain Basics

In the year 1994, much before Satoshi Nakamoto's revolutionary white paper introduced the idea of the blockchain, a well respected Computer Science Researcher and Cryptographer named Nick Szabo introduced the concept of a Smart Contract, and in 1996 wrote an entire research paper on it.

He defined smart contracts as "a set of promises, specified in digital form, including protocols within which the parties perform on these promises".

Several years after the Bitcoin white paper was released, building on the groundbreaking inventions that Szabo and Nakamoto had created, Vitalik Buterin released the Ethereum white paper titled "A Next-Generation Smart Contract and Decentralized Application Platform". This paper discussed the implementation of a blockchain protocol with a Turing-complete programming language that would allow for the creation of "complex applications involving having digital assets being directly controlled by a piece of code implementing arbitrary rules".

Buterin defines Smart Contracts as "cryptographic "boxes" that contain value that only unlock if certain conditions are met".

Meaning that, through the use of Smart Contracts, developers can define their own instructions to the Ethereum Virtual Machine, or EVM for short, allowing digitally binding agreements to be coded on a public ledger. This enables anyone to build a wide array of decentralized applications that have the capability to potentially disrupt numerous industries.

Today, with the tremendous growth of the Ethereum community, developers have several options when deciding on which Turing-complete programming language they would like to use to build their Smart Contracts and decentralized application logic.

The most notable being Solidity. Solidity is a language similar to JavaScript which allows you to develop contracts and compile to EVM bytecode. It is currently the flagship language of Ethereum. It also has the most robust documentation and is by far the most popular amongst the developers in the ecosystem.

In addition to Solidity, you have Viper, an experimental programming language that is growing tremendously in popularity. Viper was built with three objectives in mind. The first being "Security - it should be possible and natural to build secure smart contracts in Viper". The second being "Language and compiler simplicity - the language and the compiler implementation should strive to be simple", and the third being "Auditability - Viper code should be maximally human-readable. Furthermore, it should be maximally difficult to write misleading code. Simplicity for the reader is more important than simplicity for the writer, and simplicity for readers with low prior experience with Viper (and low prior experience with programming in general) is particularly important".

Another important language that is growing in popularity is Low-level Lisp-like Language, or LLL for short, which is used to write assembly code that interacts with the EVM. LLL, therefore has many unique benefits. First, the developer has direct access to memory and storage. This allows you to arrange your contract data in any way you like, optimizing for the most efficient access. Second, you have complete access to all EVM opcodes. When compiled, these LLL operators translate directly to EVM opcodes, giving your contract power and brevity. Third, and expanding on the brevity idea, LLL contracts compile down to *much* smaller binaries than Solidity contracts.

Although in this course we will focus mainly on Solidity, each of these three languages is very powerful for Smart Contract development.

Introduction to Smart Contracts -- Solidity Documentation
https://github.com/ethereum/wiki/wiki/Light-client-protocol
Hard Forks, Soft Forks, Defaults and Coercion by Vitalik Buterin

Short guide to bitcoin forks - Coindesk