**BLOCKCHAIN DATA**

**In a previous video we had an overview of the blockchain as a technology similar to a shared database. In this video we look at the structure with which data is stored on the blockchain (quote unquote) database and why this structure may be useful in aaaa casings such as crypto currencies. Blockchain, as the name suggests, stores data in a linked setting where each block of data is linked to one or more data blocks created previously, therefore making searching through the database backward in time efficient. This linkage between the data pieces is done by applying a concept that is well used in computer file searching to create what’s called Hash Pointers. Let’s take a look at this Hash concept and see why its useful. Think about a friend of yours among a room of strangers, how do you identify your friend? You might have a lot to say about them, their looks, their personality, a lot of data that’s unique to your friend that is different from the other people. However, computationally you don’t need all that data. All you need to distinguish your friend unequivocally apart from the others is a fingerprint, and this is exactly what hashing does. It takes a piece of data, of any length, and it create an almost unique fingerprint of that data of a small fixed length such that you can embed it in another piece of data linking them together. There are many hashing algorithms and one example is the SHA-256 cryptographic hashing algorithm used in the bitcoin Blockchain. Here’s how it works, take a piece of data that you want to create the fingerprint from, it could be anything text, image, video but stored on a computer this is all essentially 1s and 0s. The hashing function takes these 1s and 0s split them into chunks and runs multiple rounds of simple logical operations on them such as AND, OR, XOR etc. Essentially shuffling these 1s and 0s condensing them into fewer 1s and 0s until eventually there are only 256 1s and 0s left. That is, 256 bits 32 bites or 64 digits expressed in hexadecimal format. So essentially what hashing does is the fingerprinting process, takes some input data of any arbitrary length, compresses it to create an almost unique representation of that data in 128, 256, or other small amounts of 1s and 0s. By the way, this how searching on a computer is done. Each file is indexed, having a hash created from it and searches are done on the collection of hashes which is much smaller than the full data, making the process much easier. In order to use on a Blockchain, particularly on public Blockchains distributed to a large number of parties, a good cryptographic hash function like SHA-256 should have some good concealment and efficiency features; first it should be very quick to compute and on decent computers SHA functions usually takes a fraction of a second regardless of the data input size, second it should be very difficult to reverse engineer that is, given the hash it should be infeasible to back-out the original input. First there would need to be what’s called a high avalanche effect which means that if you change the input by just a little bit, say adding an additional 0, the new hash should be dramatically different from the original hash, this ensures that you can’t simply guess the input by pattern analysis. In fact, the only way to guess the input would be ‘brute forcing’, trying all the input hashing one at a time to find the input that produces that hash. Taken together these features ensures that a cryptographic hashing function is essentially a one-way function meaning it’s easy to compute but very hard to reverse. This property is very important and is the foundation of a lot of applications where it’s important to check the authenticity of some data, but you don’t want to necessarily reveal that data. One example is password storage. The password associated with your online account is stored on the server in a hash, so even if someone breeches the site they still can’t easily reverse it unless they ‘brute force’ but it is easy to check because when you enter the password it is hashed and the hash is compared with the stored copy granting you access if these match.**

**The other example is the bitcoin mining puzzle which we will talk about in the next module. Here in a generic Blockchain setting, hashing is primarily used to create the data fingerprint allowing us to link the data together. And here’s how it works. In the beginning some data is generated and stored on the blockchain, this data block is known as the genesis block as it marks the start of the chain. When the next batch of data is created and stored however, it will include a hash of the previous block. This unique fingerprint essentially creates a pointer pointing towards the previous block and it’s the same process for future blocks as well. The next block will contain a hash of the pervious block which itself contains the hash of the block before and so forth. Note that we can use these hash-pointers anywhere, not just on the blocks itself but individual data pieces within each block as well. For instance, the Bitcoin Blockchain stores individual transaction data and each transaction could contain the hash of one or more previous transactions showing the previous receipt of the coins, this is the chain concept in Blockchain. Hash-pointer allows data blocks and individual data pieces to be chained to previous histories. The reason this particular structure is needed is two-fold 1st just like any file search hashing makes searching through the Blockchain easier. To locate some particular data you just need to take a hash of it and search through all the previous hashes stored on the Blockchain which is much smaller in size. And, this is particularly important in a distributed Blockchain like Bitcoin where searches are done over 10s and 1,000s of nodes, in addition hash pointers provides a tamper-evident data log on the Blockchain which again on a public setting serves to enhance data integrity. To see this, suppose someone tries to alter the historical data in say… block one, maybe changing the ownership of some coins, but this will be immediately evident to all Blockchain nodes because the altered block when hashed won’t match the historical hash of that block in block 2, which again when hashed won’t match the stored hash in block 3 and so on. Therefore, attempts to manipulate the data on a well distributed Blockchain will be quite obvious and the proof of any action is evident using hashes. Notice that I labeled this as tamper-evident and not tamper-proof or data immutability and its important to distinguish these claims as they are often used interchangeably in marketing pitches. Here the hash-pointers makes tampering attempts evident to all nodes of the network, tampering is still very possible especially if the tampering party happens to control a large number of nodes using say… a 51% attack. We’ll look at this distinction in more detail in the video on consensus algorithms.**