Blockchains Architecture, Design and Use Cases

Prof. Sandip Chakraborty
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

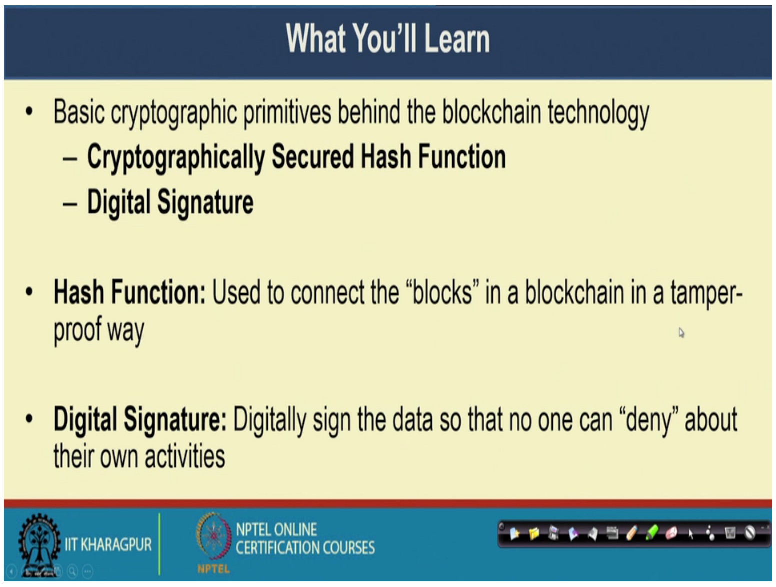
Lecture – 05
Basic Crypto Primitives – I

So, welcome to all of you now once again under course on blockchain, where we are discussing the basic architecture of blockchain and its usefulness. So, in the last lectures we have discussed about the fundamentals of blockchain and how this blockchain technology can be applied in wide for different kind of application ranging from financial applications as well as different types of applications like smart contracts or for different types of business platforms.

So, with that background; now we will start the discussion of in depth analysis of the blockchain technology its design principle and how we can use blockchain for different type of applications starting from the details of the bitcoin technology. So, initially we will look into the basic primitives, which works as the building block behind the development of a blockchain technology and the bitcoin architecture. And then from there gradually we will look into the into the various things inside the broad concepts of bitcoin architecture and how bitcoin is developed as an useful technology, for money transfer or the digital currency transfer over a broad network.

So, let us start with the basic things behind the development of behind the development of cryptographical aspects of a blockchain. So, we will discuss broadly about the basic crypto primitives in next couple of lectures, initially will look into the details of hash function and from there we will go to the details of digital signatures.

(Refer Slide Time: 02:18)



What You'll Learn

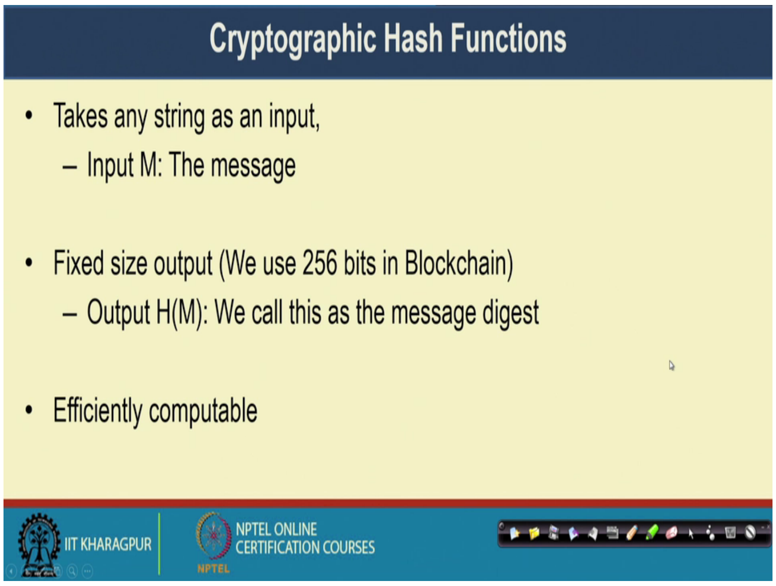
- Basic cryptographic primitives behind the blockchain technology
  - **Cryptographically Secured Hash Function**
  - **Digital Signature**
- **Hash Function:** Used to connect the "blocks" in a blockchain in a tamper-proof way
- **Digital Signature:** Digitally sign the data so that no one can "deny" about their own activities

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

So, in this set of lectures we will look into the broadly two concepts, one is cryptographically secured hash function and the second concept of digital signatures, which work as the fundamental building block behind the blockchain technology. So, this hash function they are used to connect different blocks one after another that we had seen earlier during the basic discussion the that, individual blocks of a blockchain they are connected with each other through some hash function.

And then we will look into the concept of digital signature, where this digital signature is used for making the blockchained blockchain tamper proof and resilient against non-reparation kind of attack, where once party makes a transaction he or she will not be able to deny that the transaction has not been made or he or she has not made that transaction.

(Refer Slide Time: 03:24)



Cryptographic Hash Functions

- Takes any string as an input,
  - Input M: The message
- Fixed size output (We use 256 bits in Blockchain)
  - Output H(M): We call this as the message digest
- Efficiently computable

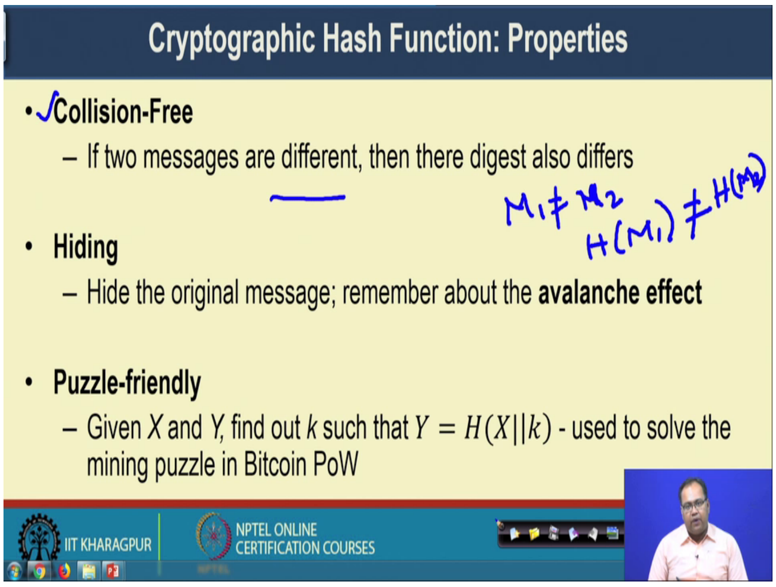
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, let us start with this concept of cryptographic hash function, in the last lecture we have seen deep details of the cryptographic hash function. So, a cryptographic hash function it takes an input which is the message and it produce an output H M which we call as the message digest. Now, in case of black blockchain, this message digest is a fixed size output and we generally use 256 bits of outputs in blockchain.

So, the property of this kind of cryptographic hash function is that this kind of hash functions are efficiently computable function and you do not need to use a significant amount of resource just to be sure that the message digest of a message in corresponds to H M, where the algorithm corresponds to the hash is known.

So, from this point the three important properties of a cryptographic hash functions are as follows.

(Refer Slide Time: 04:34)



Cryptographic Hash Function: Properties

- ✓Collision-Free
  - If two messages are different, then there digest also differs

$M_1 \neq M_2$

$H(M_1) \neq H(M_2)$

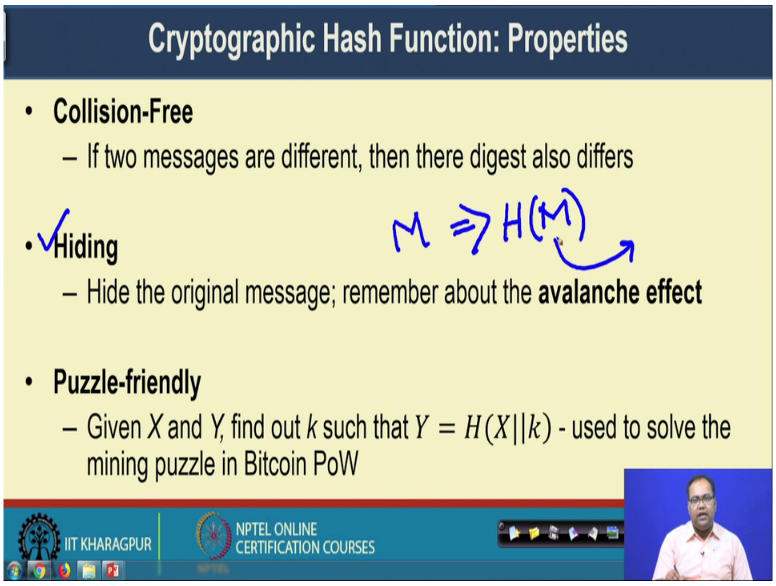
- Hiding
  - Hide the original message; remember about the **avalanche effect**
- Puzzle-friendly
  - Given X and Y, find out k such that $Y = H(X||k)$ - used to solve the mining puzzle in Bitcoin PoW

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

First the hash function need to be collision free. So, called by collision free we means that if 2 messages are different, then their digest will also differ. That means, that if you have 2 messages M 1 and M 2 and if you know that M 1 is not equal to M 2 then their digest H of M 1 will also not match with H of M 2.

So, that is the basic fundamental concept behind is collision free property of a cryptographic hash function, the second Intel important property for a hash function is hiding. So, this kind of hash function it hide the original message behind the message digest.

(Refer Slide Time: 05:21)


Cryptographic Hash Function: Properties

- **Collision-Free**
  - If two messages are different, then there digest also differs
- ✓**Hiding**
  - Hide the original message; remember about the **avalanche effect**
- **Puzzle-friendly**
  - Given X and Y, find out k such that $Y = H(X||k)$ - used to solve the mining puzzle in Bitcoin PoW

$M \Rightarrow H(M)$

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, whenever you are transferring H M, from H M it is very difficult to guess what is the inherent message M. By given a M you can efficiently compute H M, but given H M there is no efficient algorithm exist which can compute M. So, that way we call this kind of hash function as the one way function, that given an M given a message you can efficiently compute the corresponding digest, but given a digest there does not exist any efficient algorithm which can compute the original message or the M. So, that is the property of a message hiding.

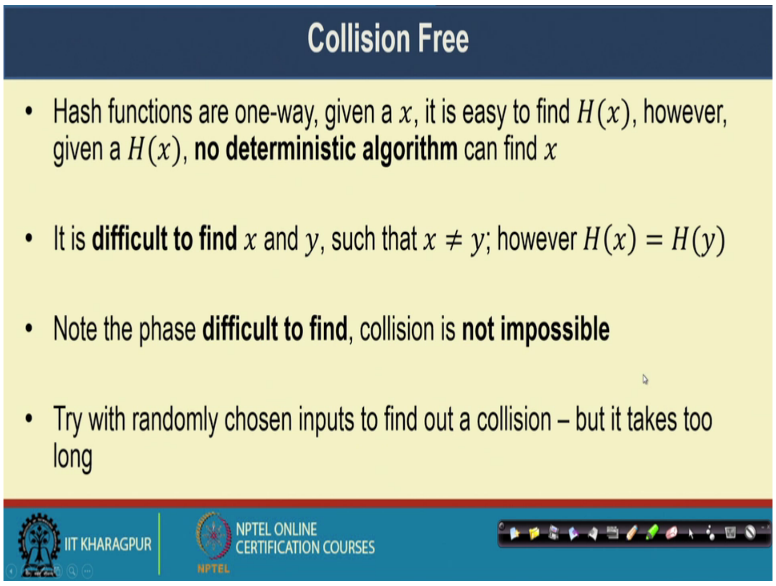
And the third interesting property which is used in the blockchain design and which is another interesting property behind the development of cryptographic hash function, it is called the puzzle friendly property. So, the puzzle friendly property says that given 2 messages X and Y, you need to find out a value k such that y will be equal to hash of x appended k.

So, this means that you need to find out the value of k, where X is given and Y is given and the algorithm for the hash is also given. So, for this kind of methodology or for this kind of puzzle, there does not exist any efficient algorithm which can compute k in a in a very efficient way. So, that way if you ask any person to compute suck k where, X and Y are given, then the best way to solve this kind of puzzle is by applying random methodology.

So, you have to try for different values of k and you have to find out whether the hash of X appended k becomes equals to Y.

So, these are the three important properties of a cryptographic hash function.

(Refer Slide Time: 17:15)



Collision Free

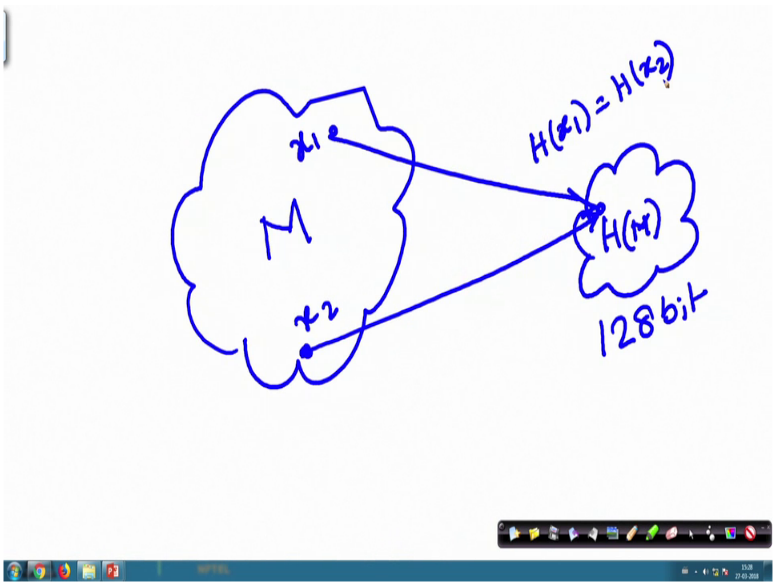
- Hash functions are one-way, given a x, it is easy to find $H(x)$, however, given a $H(x)$, **no deterministic algorithm** can find x
- It is **difficult to find** x and y, such that $x \neq y$; however $H(x) = H(y)$
- Note the phase **difficult to find**, collision is **not impossible**
- Try with randomly chosen inputs to find out a collision – but it takes too long

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

So, we look into this three properties little details. So, as I have mentioned that the collision free property says that this hash functions are one way. So, given some value of x it is computationally easy to find the H x value of a H x, but given a h x you cannot have any computationally efficient way to find out the original message. So, from message to digest you can find out it using an efficient mechanism, but from digest to message there is no such efficient way to find it out.

So, that particular property we call it as collision free and one interesting point here to note is that, we have we have mentioned that such kind of things like it is it is difficult to find the values of X and Y such that X not equal to Y. However, H x equal to H y so; that means, there is a kind of collision in the hash and as we have discussed earlier that this kind of collision can always occur in a hash function, because in a hash function typically what we are doing that.

(Refer Slide Time: 08:26)



You have a large message pool, which is the given M and from that large message pool you have the corresponding digest and the digest has some fixed size say for the digest has some 128 bit of length.

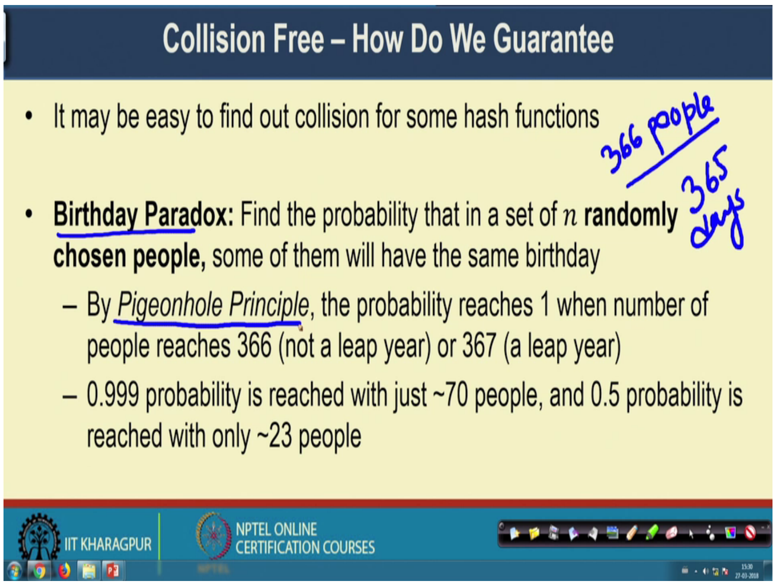
Now, whenever you are mapping a large population to a corresponding small population, it is always like that there would be 2 point here where that they will be mapped to the same point in H M. So, this can always be there the case, but it is it is very unlikely that an attacker, he will be able to find out 2 such x 1 and x 2 where H of x 1 becomes H of x 2. So, this is kind of difficult to find in case of a cryptographic hash function.

So, what I mentioned here that this kind of property where x is not equal to y, but H of x is equal to H of y this is difficult to find, but as I have mentioned it is not impossible to find out such kind of computation and the best way of doing this kind of thing is to try randomly chosen input, and to find out whether a collision occur in case of a hash function. But remember that there is no such efficient algorithm exist, which can find out a collision in a cryptographic hash function.

So, let us see that how do we guarantee this kind of collision proper property and as I you have mentioned again I am repeating the same point that, this kind of collision is difficult to find in case of a hash function. Now, for certain hash function it depends on the design of the hash function, for certain hash function it may be easy to find out such kind of collision, but for certain other hash function it may be difficult to find out. And with the term cryptography hash function, we are talking about the hash function where such kind of collision is difficult to find.

Now, let us look this property from a probabilistic way that we need can find out that given some fixed length of the population of the possible message digest, what is the probability that you will be able to find out to such messages, where they will be mapped to the same digest.

(Refer Slide Time: 10:56)



Collision Free – How Do We Guarantee

- It may be easy to find out collision for some hash functions
- Birthday Paradox: Find the probability that in a set of n **randomly chosen people,** some of them will have the same birthday
  - By *Pigeonhole Principle*, the probability reaches 1 when number of people reaches 366 (not a leap year) or 367 (a leap year)
  - 0.999 probability is reached with just ~70 people, and 0.5 probability is reached with only ~23 people

366 people
365 days

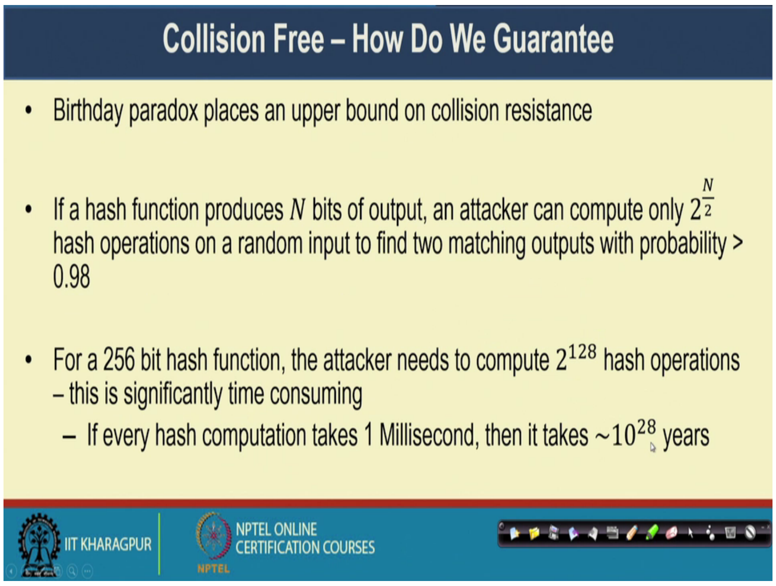
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, this kind of a problem we can solve using the concept of birthday paradox. So, the birthday paradox says that find the probability that in a set of n randomly chosen people, some of them will have the exactly same birthday.

Now, in this birthday paradox you see that if you have 366 number of people in the population, if you consider a leap year then there is certainly a collision. Why because you know that in a year which is not a leap year, you have a total of 365 days and out of this 365 days, you can have 365 possible birthdays. And if you have 366 people in the population then it is guaranteed by the pigeonhole principle that 2 of them will have the exactly same birthday.

So, that way whenever the population reaches to 366 if it is not a leap year or equal to 367 if it is a leap year, then it is guaranteed or with probability one you will be able to say that there is a collision. But interestingly if you if you just try to see that what will happen in case of a list number of people compared to 366 for a leap year or 367 367 for a leap year or 366 for not a leap year, that around 0.999 probability; that means, 99.9 percent probability is reached with just 70 percent 70 people. On the other hand you can get 0.5 probability with only 23 people; that way you will see that if you have some sufficiently random pick up of small number of persons from the population with high probability will be able to say that the 2 persons will have same birthday.

(Refer Slide Time: 12:59)



Collision Free – How Do We Guarantee

- Birthday paradox places an upper bound on collision resistance
- If a hash function produces N bits of output, an attacker can compute only $2^{\frac{N}{2}}$ hash operations on a random input to find two matching outputs with probability > 0.98
- For a 256 bit hash function, the attacker needs to compute 2^{128} hash operations – this is significantly time consuming
  - If every hash computation takes 1 Millisecond, then it takes $\sim 10^{28}$ years

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

Now, this kind the property also applies in the concept of cryptographic hash function to find out a hash collision. So, this birthday paradox it gives a upper bound on the collision resistance.

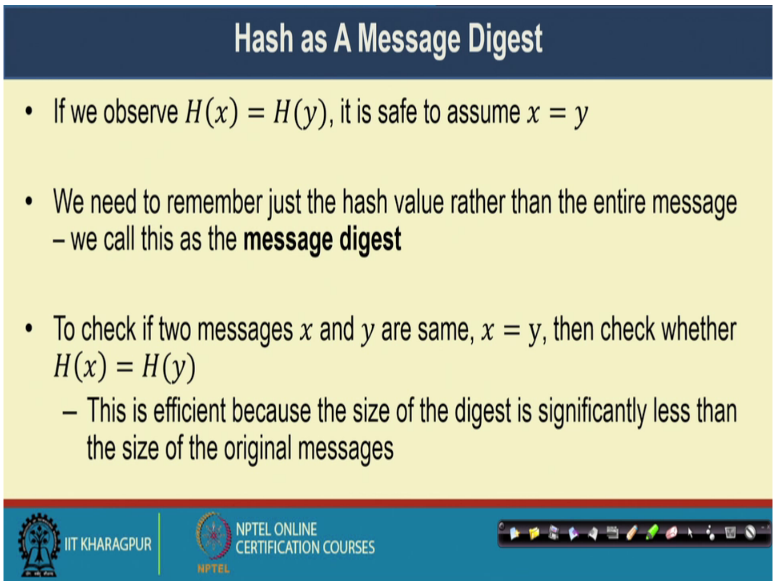
So, if a hash function produces n bits of output, then we can say that an attacker if it if that attacker can compute 2 to the power n by 2 hash operations on a randomly input randomly chosen input, to find out that whether 2 match whether there is a hash collision; that means, whether there are 2 matching outputs, it can find it out with a probability greater than 0.98.

That means, with 98 percent surety, the attacker will be able to say that or attacker will be able to find out that they are exist to messages which maps to the same message digest and that attacker need to find out only 2 to the power n by 2 number of possible hash operations.

Now, the question is that for a 256 bit output, the attacker need to compute 2 to the power 128 such hash operation. So, we need to see whether this is feasible or not. Now if you look into a small mathematics like if every hash computation takes one millisecond, then you will require 2 to the power 128 millisecond to compute 2 to the power 128 numbers of hash operations and which is approximately equal to 10 to the power 28 years. So, it will take a significant amount of time to find out 2 messages which will map to the message digest.

Now, with this concept we can say that although it is not possible or although it is possible to find out a collision in a cryptographic hash function, but it is very difficult to find out 2 messages, which will map to the same message digest for a cryptographic hash function. So, that was the first property of this collision free then we this collision free property we can say that if 2 hashes or 2 message digest are similar.

(Refer Slide Time: 15:10)



Hash as A Message Digest

- If we observe $H(x) = H(y)$, it is safe to assume $x = y$
- We need to remember just the hash value rather than the entire message – we call this as the **message digest**
- To check if two messages x and y are same, $x = y$, then check whether $H(x) = H(y)$
  - This is efficient because the size of the digest is significantly less than the size of the original messages

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

So, if 2 message does digest are similar, then we can safely assume that the corresponding messages are also similar by considering that it is it is very difficult to find out 2 messages, which will map to the same digest.

Now, with this particular property, we can claim that if you want to transfer some message then if you want to check the validity of that message, you just need to remember the hash value not the entire message. So, that way we call this particular hash value of a message as the message digest that you do not need to remember the entire message to check whether 2 messages are similar or not, you can just compute the corresponding hash function remember the message digest and by comparing the 2 message digest if the message digest become similar, then with high probability you will be able to say that the corresponding messages are also similar.

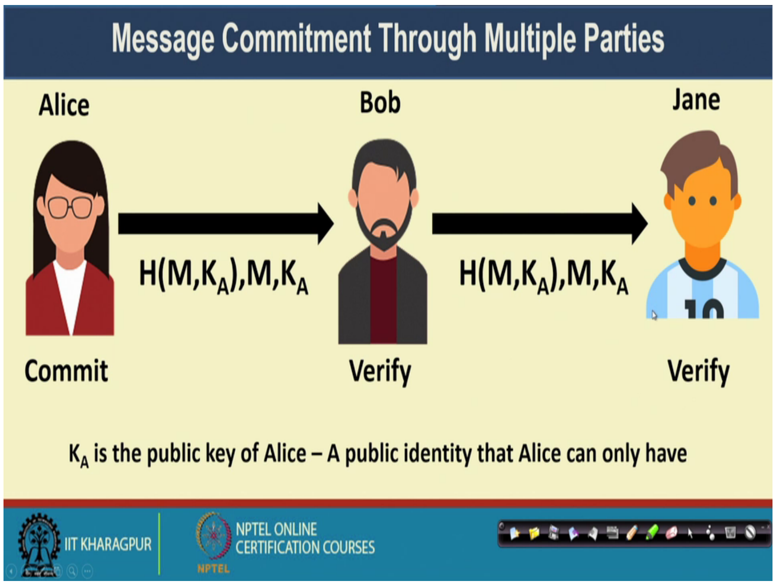
So, this particular computation is efficient, because the digest of the size of the digest is significantly less compared to the size of the message size of the original message. So, you do you do not need to do a comparison over large number of bits, you can just make a comparison over the message digest, which are a finite bit size. So, of a 256 bit message digest you just need to do a 256 bit wise comparison to find out whether 2 message are same or not.

So, the second properties of a second important property of a cryptographic hash function, is the information hiding property. So, as you have mentioned earlier that this hash functions are one way function; that means, it is computationally difficult to find out the original massage, if the message digest is given to you. And it also defect and it is difficult and a difficulty depends on the size of the message digest.

So, if I give you or a 2 bit message digest, it may be easier to find out what is the original massage, but if I give you a 256 bit message digest, it becomes very difficult for you to guess what is the original message. So, this particular one way property of a hash function it helps hiding the message. Hiding the message in the sense like you can you can hide the original message behind the message digest and the advantage is that with the message digest you can make a comparison whether 2 messages are similar or not.

Now, this kind of hiding it helps to commit a value and check it later. It is like that you can you can compute the message digest compare corresponds to a message store the message digest along with the message, and next time you got another message you want to compare that message with the original message. So, you compute the message digest and then check whether 2 message digest are same. You do not need to check the original messages with each other.

(Refer Slide Time: 18:26)


Message Commitment Through Multiple Parties

K_A is the public key of Alice – A public identity that Alice can only have

This gives an interesting application in the direction of message commitment.

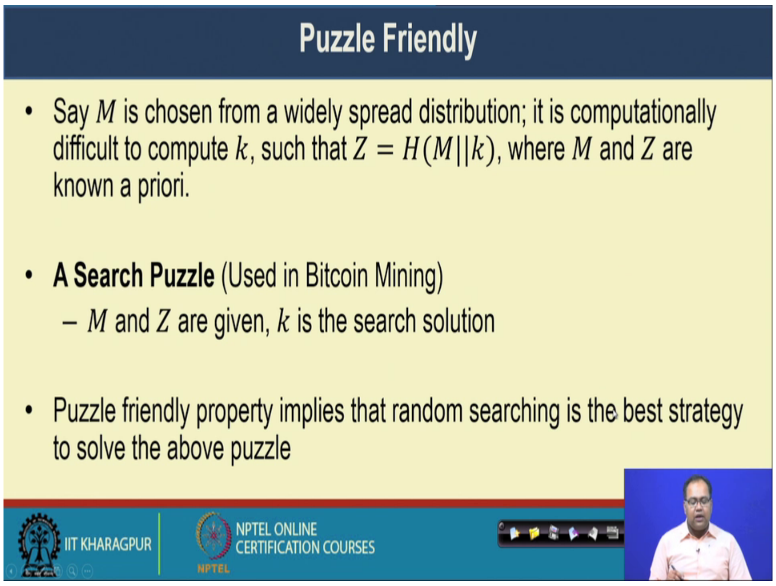
So, in this particular application Alice wants to send some information to Jane via Bob. So, it is like that now Alice is sending this information to Bob and whenever Alice is sending this information to Bob what Alice is doing; that Alice is sending the message along with that Alice is sending one information, which we called the public key of Alice.

Now it is globally known that this public identity Alice can only have this public identity K A. So, if you if you have the it is information K A, you can safely assume that this information K A is known to Alice only or that belongs to Alice only. Now, Alice sends this information along with M and along with that it sends a hash value. So, this hash value combines the original message and a secret information or the information that Alice has and make a hash of done.

Now, whenever you are sending this information via Bob, if Bob makes any changes in the message Bob has to also compute the corresponding things corresponding hash. Now this particular thing Bob cannot compute because this information is known to Alice only.

So, the details of this thing we will look into later, but the idea is that this Jane it can verify whether this intermediate person has made any changes in the message or not, by computing the hash value from these two parameter and then checking the hash value with the hash value that was transferred by Alice originally. So, if Bob makes any changes in the value of M. So, that will also make a change in H of M K A. So, it will it will get verified during the commitment of commitment of this kind of messages.

(Refer Slide Time: 20:34)



Puzzle Friendly

- Say M is chosen from a widely spread distribution; it is computationally difficult to compute k, such that $Z = H(M||k)$, where M and Z are known a priori.
- **A Search Puzzle** (Used in Bitcoin Mining)
  - M and Z are given, k is the search solution
- Puzzle friendly property implies that random searching is the best strategy to solve the above puzzle

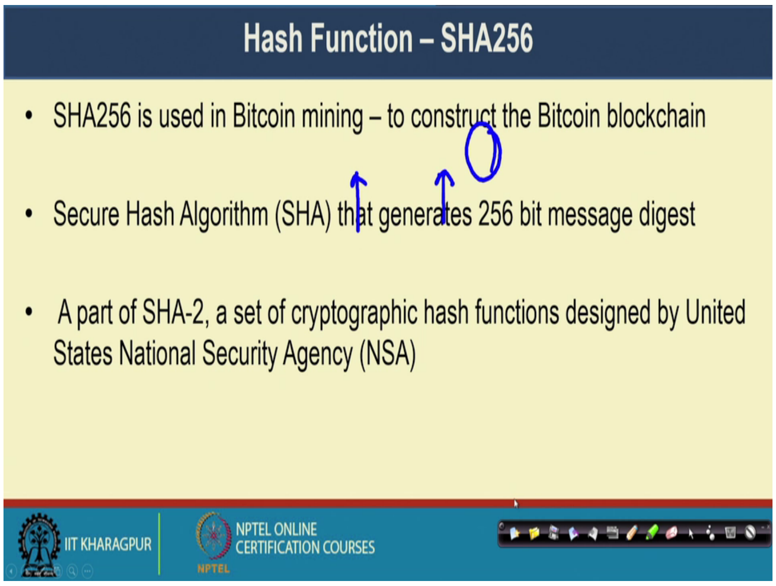
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

Now, that hard property of a cryptographic hash function is puzzle friendly property. So, this puzzle friendly property you have mentioned earlier; in case of a puzzle friendly property it says that you have been given you have been given M and Z. And M and Z is given to you and you need to find out the value of k such that Z becomes equal to the H of M and k append at with each other.

Now as you have mentioned earlier that finding out such k is extremely difficult in case of a cryptographic hash function and the best way to solve this puzzle is to randomly try with different values of k and check that whether the hash function results the value of Z or not. So, this corresponds to the puzzle free property. So, the puzzle free property implies that you need to make a random searching of the parameter k, to find out the hash value of M and k appended together and hash value of M and k with becomes equal to Z.

So, this puzzles friendly property helps us a significantly in the construction of a blockchain architecture.

(Refer Slide Time: 21:54)



Hash Function – SHA256

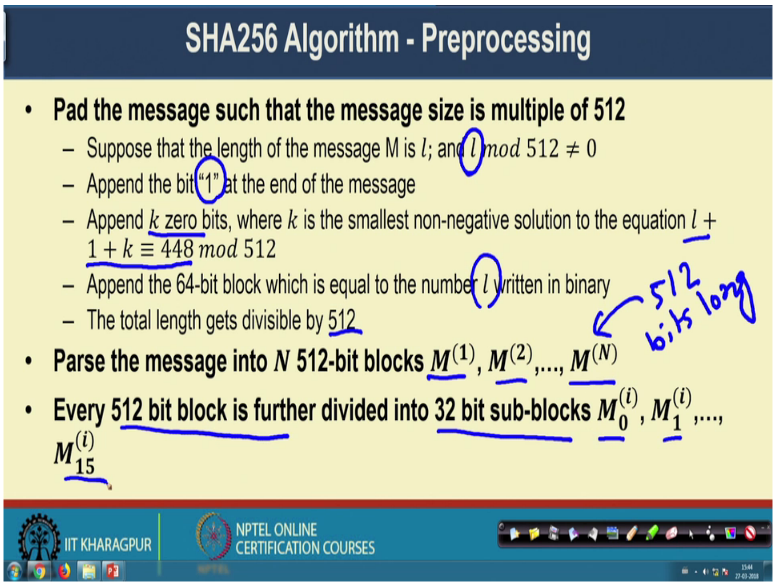
- SHA256 is used in Bitcoin mining – to construct the Bitcoin blockchain
- Secure Hash Algorithm (SHA) that generates 256 bit message digest
- A part of SHA-2, a set of cryptographic hash functions designed by United States National Security Agency (NSA)

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

So, let us see how it helps us in solving the blockchain. So, in case of a blockchain we use a special type of hash function called SHA 256 hash function. So, this SHA 256 hash function it is used during the Bitcoin mining phase to construct the blockchain behind Bitcoin. So, SHA corresponds to the secure hash algorithm which generates 256 bit message digest.

So, the output generated by this SHA 256 hash algorithm is a 256 bit message digest. So, this hash function it is a part of SHA 2 group of hash functions, which is which are a set of cryptographic hash function, which are designed by United States National Security Agency; so, it belongs to different kinds of hash function like SHA 128, SHA 256, SHA 512. So, out of that that SHA 256 is used during the Bitcoin mining procedure.

(Refer Slide Time: 23:03)



SHA256 Algorithm - Preprocessing

- **Pad the message such that the message size is multiple of 512**
  - Suppose that the length of the message M is l; and l mod $512 \neq 0$
  - Append the bit "1" at the end of the message
  - Append k zero bits, where k is the smallest non-negative solution to the equation $l + 1 + k \equiv 448$ mod 512
  - Append the 64-bit block which is equal to the number l written in binary
  - The total length gets divisible by 512
- **Parse the message into N 512-bit blocks $M^{(1)}, M^{(2)}, \ldots, M^{(N)}$**
- **Every 512 bit block is further divided into 32 bit sub-blocks $M_0^{(i)}, M_1^{(i)}, \ldots, M_{15}^{(i)}$**

So, let us loop briefly about this SHA 256 algorithm the hash algorithm. So, it uses a preprocessing mechanism in this preprocessing mechanism, the faster case you need to pad the message such that the total me message size becomes multiple of 512 bits. So, it may it may happen that the total message size is less than an integer multiple of 512 bits, if it is in less than an integer be multiple of 512 bits, then you have to at certain pad padding bits with that and the padding bits which are added is first you have to append bit one at the end of your message.

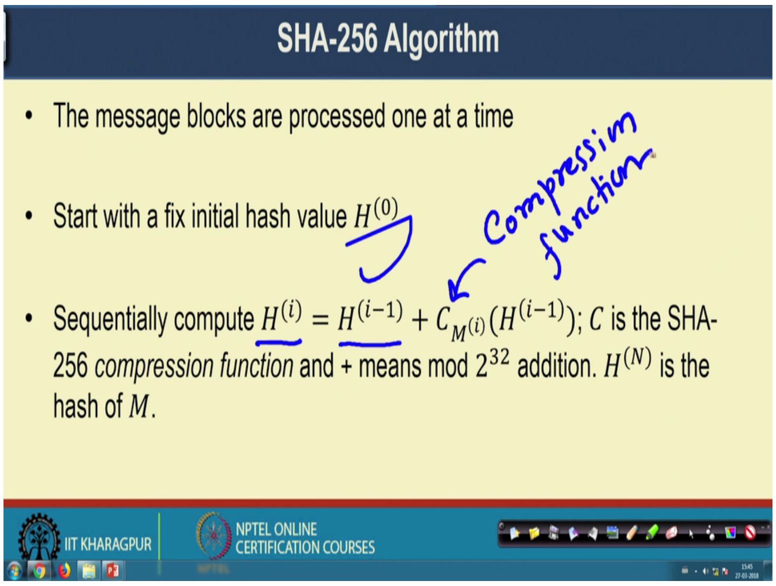
After appending a bit 1 at the end of your message you have to append k number of zero bits, where this value of l, l is the value of your original message length. So, your original message was l bit and our objective is to ensure that l mod 512 becomes equal to 0 and they are you first pad a 1 bit, after that you pad k and the zero bits such that this l plus 1 plus k becomes equivalent to 448 mod 512.

Now, whenever you are getting such k number of bits, which are being appended, the total message size becomes integer multiple of 448. So, you have to append 64 more bits. So, this 64 bit block, which is appended at the end to make it an integer multiple of 512 it is the binary equivalent of this value l. So, if you convert the value l to binary, whatever number of bits you get in a 64 bit representation, that is appended at the end and after appending that at the end you get total length block which is an integer multiple of 512.

Now, once you have this total length integer multiple of 512 by adding updates extra padding bits, you pass this message into n number of 512 bits block. Now, the total message length has become integer multiple of 512 bits. So, you can divide the message into multiple chunks like M 1 M 2 up to M n such that each such message block is 512 bit long. After that each of these 512 bit message they are further divided into 32 bit sub blocks M 0 i M 1 i to M 15 i.

So, after doing this preprocessing means after dividing this entire message into multiple blocks the next task, the next task is to start with a fix initial hash value of H 0.

(Refer Slide Time: 25:59)



SHA-256 Algorithm

- The message blocks are processed one at a time
- Start with a fix initial hash value $H^{(0)}$
- Sequentially compute $\underline{H^{(i)}} = \underline{H^{(i-1)}} + C_{M^{(i)}}(H^{(i-1)})$; C is the SHA-256 *compression function* and + means mod 2^{32} addition. $H^{(N)}$ is the hash of M.

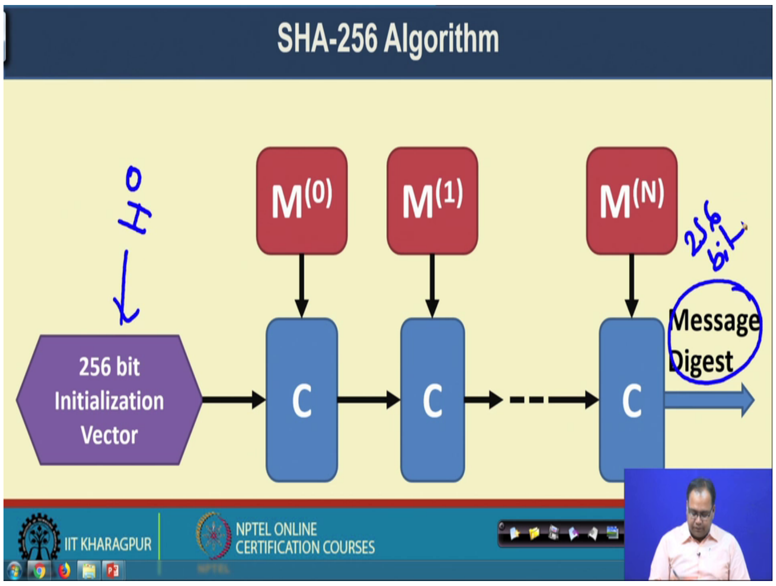
Compression function

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, you start with a hash value of 1 0, after starting with a hash value of 1 0, you iteratively process each block one after another. So, you sequentially compute this H i which will be H of i minus 1 plus a compression function. So, this is a compression function we are not going to the details of how this compression function looks like, if you are interested you can look the details of this compression function. So, this compression function is a sequence of shifting bit shifting and bit appending operations.

So, you apply this compare compression function over the previous hash function that you have obtained, and you with the message block that you have and by after applying the compression function you add it up with. So, this add is the bitwise and operation, add it up with the original hash value at the i plus 1 the iteration and that way you do the computation and the at the end whatever you will get that will give you the final hash value. So, an example is something like this.

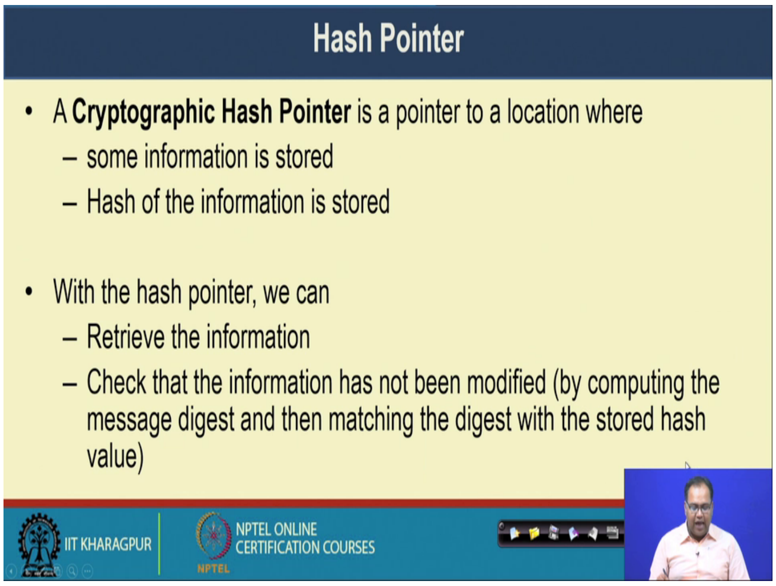
(Refer Slide Time: 27:19)



So, you have a 256 initialization vector. So, these 256 C C initialization vector was your initial hash value or H 0. Now these 256 initial vector is given as a input to the compression function and the first block is given to input, then the compression function produces the output. With this output you are then taking the input of the next block applying at the compression function and then doing the addition with the previous hash value sending it out.

And after that finally, the hash value that you will get from here that will be equal to your message digest. Now, initially you have taken a 256 bit initial vector and while doing this compression function the compression function again generates a 256 bit output and whenever this things are getting added sequentially finally, you will get the message digest which is 256 bit long.

(Refer Slide Time: 28:17)



Hash Pointer

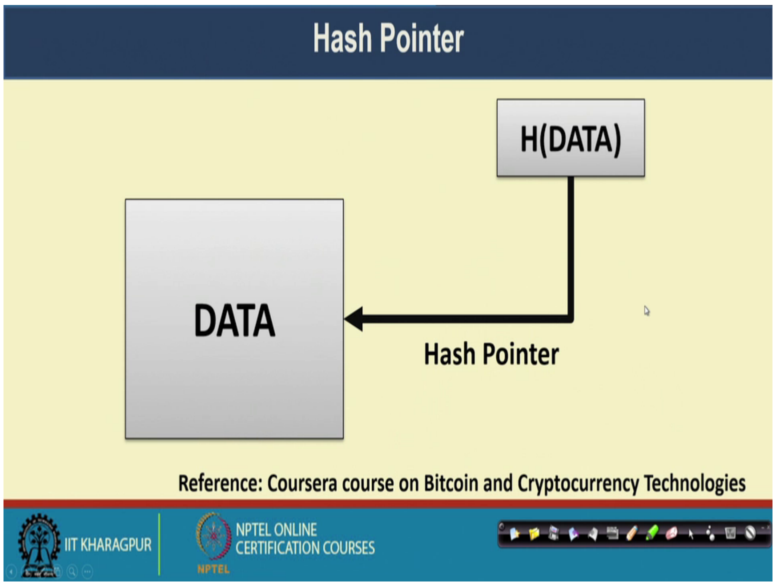
- A **Cryptographic Hash Pointer** is a pointer to a location where
  - some information is stored
  - Hash of the information is stored
- With the hash pointer, we can
  - Retrieve the information
  - Check that the information has not been modified (by computing the message digest and then matching the digest with the stored hash value)

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, this is the SHA 256 algorithm, now we come to the concept of hash pointer which we use in the blockchain concept. So, cryptographic hash pointer it is a pointer to a location, where is total original message are the information along with the hash of that information.

So, with this particular hash pointer you can retrieve the information and you can make a sanity check, that whatever message that has been stored in the location. If the hash value matches with the message digest then this particular message digest actually points to the original message.

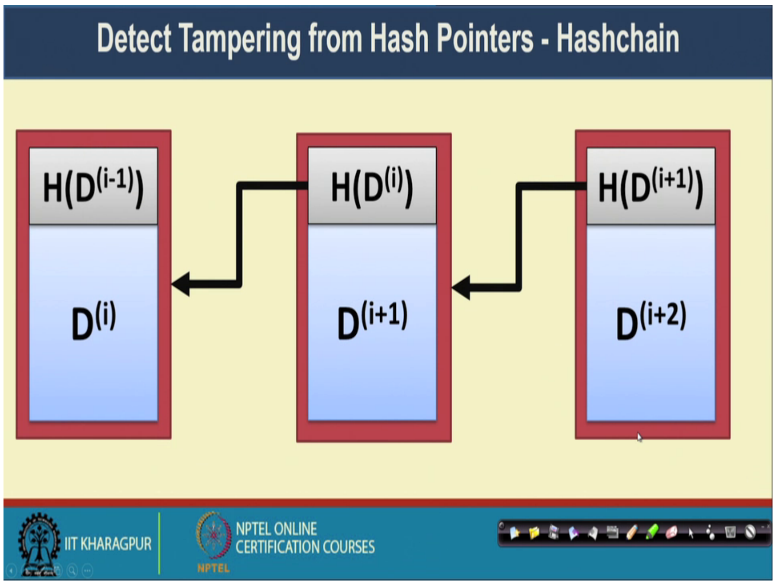
(Refer Slide Time: 28:53)


Hash Pointer

Reference: Coursera course on Bitcoin and Cryptocurrency Technologies

So, this is an example of a hash pointer. So, in this hash pointer you have the original data that was there, and the hash value of the data it is pointing to the original data block.

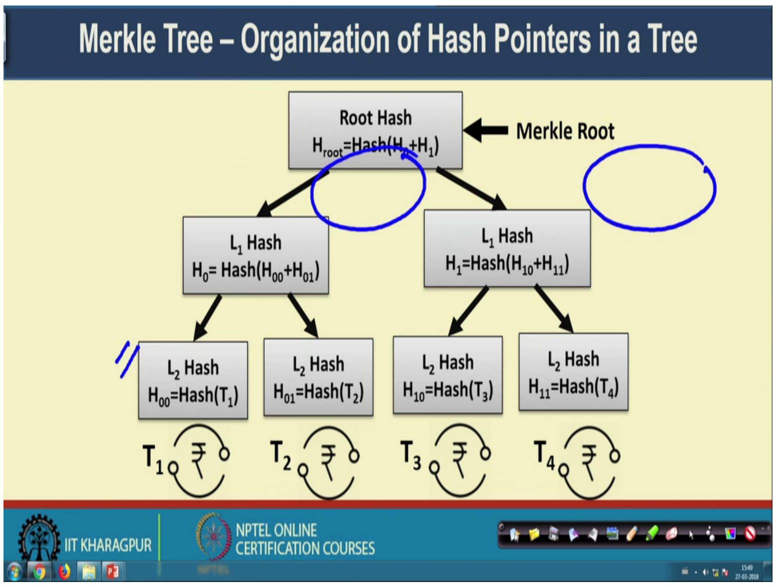
(Refer Slide Time: 29:11)


Detect Tampering from Hash Pointers - Hashchain

Now, with this concept of hash pointer you can detect tampering. So, we use this kind of data structure in blockchain construction, what we call as the hashchain. So, in this hashchain architecture this particular hash value it points to the previous data block. So, these hash value of this D i the data block along with the previous hash value, that is getting hash and the original hash value is appended to the data block of data block of i plus 1 data block.

So, that way if you are making any change in this data blocked, then you need to make a change at this particular hash value and if you are making this changes this particular hash value, then this particular hash value will also get changed and that way if you want to make change in one of the particular data block, it will have a kind of cascading effect that all subsequent hash values need to be changed.

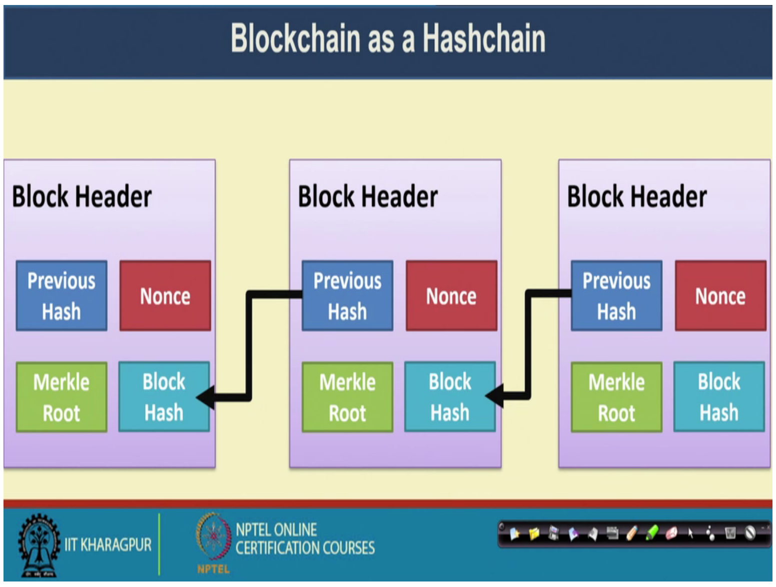
(Refer Slide Time: 30:19)


Merkle Tree – Organization of Hash Pointers in a Tree

So, this particular data structure we call it as a hashchain and best on this hash pointer you have another nice data structure, which is called the merkle tree that we have looked earlier. So, in case of merkle tree the root hash has 2 pointers, the left pointer points to the level one hash, the right pointer point to the right hash and just the nodes immediately before the leaf node, this points to individual hash of individual transaction.

So, that way if you make any changes in one of the transaction, it gets reflected in the entire path and that way we will be the entire structure is tamper proof; that means, if you make any change in one transaction, you need to make change in all the transactions along the path.

(Refer Slide Time: 31:16)



Now, as I have mentioned that this blockchain architecture it relies on this kind of hash properties. So, here using this hash pointers we are pointing, the previous hash it is pointing the hash value of the previous block and here we keep this information inside the block, this things we have discussed earlier just a kind of repetition you have a merkle root.

So, this merkle root it is the root of the merkle tree that is constructed over the set of transactions and then you have a nonce value the minor need to find out this nonce value to compute the hash of this block. So, every block will also contain the previous hash. So, that way if you want to make any changes in any of the transaction, that will get make a change in the merkle root and if you make a change in the merkle root this block hash will get change.

If this block hash get change you have to make a change in the hash value here and that way you will have a cascading effect. And if you make any changes in any of the blocks that you need to make changes in all the blocks in the blockchain.

So, this gives an idea about how we apply this concept of hash cryptographic hash function to derive the entire data structure corresponds to a blockchain, and how we made the blockchain tamper free by applying this concept of repetitive hash pointers. In the in the next lecture we will look into another cryptographic formulation called digital signature, with further helps securing the blockchain architecture. So, see you see you all during the next class.

Thank you.