**Lecture 5 Notes**

Slide 4

**Deterministic**

* means that, given the same input, it will always produce the same output. This sounds pretty obvious, but there actually are hash functions that are probabilistic.
* And there are even some cryptographic uses for them. But the purposes we have been discussing require determinism.
* Although not a top priority, we do need it to be fast. We don't want this to become the bottleneck in our ability to carry out high speed communications.
* But in general, we're usually willing to, and usually have to, sacrifice a lot of speed before we start giving up performance in the other areas.
* As a result, cryptographic hash functions are commonly hundreds of times slower than their non cryptographic peers.
* But in addition to raw speed, we also want the hash function to be computationally simple.
* So that it can be implemented in hardware and run on slow, memory starved devices such as the chips used in credit cards.

**Pre-Image Resistance**

* This property means that it should be computationally hard to reverse a hash function.
* In other words, if a hash function h produced a hash value z, then it should be a difficult process to find any input value x that hashes to z.
* This property protects against an attacker who only has a hash value and is trying to find the input.

**Collision Resistance**

* This property means it should be hard to find two different inputs of any length that result in the same hash. This property is also referred to as collision free hash function.
* In other words, for a hash function h, it is hard to find any two different inputs x and y such that h(x) = h(y).
* Since, hash function is compressing function with fixed hash length, it is impossible for a hash function not to have collisions. This property of collision free only confirms that these collisions should be hard to find.
* This property makes it very difficult for an attacker to find two input values with the same hash.
* Also, if a hash function is collision-resistant **then it is second pre-image resistant.**

**They should be puzzle-friendly**.

* It should be difficult to select an input that provides a pre-defined output. Thus, the input should be selected from a distribution that's as wide as possible.
* Cryptographic hash functions need to be deterministic, uncorrelated, efficient, preimage resistant, second preimage resistant, and collision resistant. No real hash function is likely to achieve all of these goals, and instead are going to represent compromises between them. Unfortunately, the nature of the compromises are often such that the implications of how much we are giving up in the other areas often are not really known, which leaves the door open for Mallory and his friends.

Slide 6

**Pointers**

* Pointers are variables in programming which stores the address of another variable. Usually normal variables in any programming language store data.
* Eg. int a = 10, means that there is a variable “a” which stores integer values. In this case, it is storing an integer value which is 10. This is a normal variable.
* Pointers, however, instead of storing values will store addresses of other variables. Which is why they are called pointers, because they are literally pointing towards the location of other variables.

**Linked Lists**

* A linked list is one of the most important items in data structures. This is what a linked list looks like:
* It is a sequence of blocks, each containing data that is linked to the next block via a pointer. The pointer variable, in this case, contains the address of the next node in it and hence the connection is made. The last node, as you can see, has a null pointer which means that it has no value.
* One important thing to note here, the pointer inside each block contains the address of the next block. That is how the pointing is achieved. Now you might be asking what does that means for the first block in the list? Where does the pointer of the first block stay?
* The first block is called the “genesis block” and its pointer lies out in the system itself. It sort of looks like this:

Slide 7

* **A block header contains:**
  + - Version: The block version number
    - Time: the current timestamp
    - The current difficulty target
    - Hash of the previous block
    - Nonce
    - Hash of the Merkle Root