* The only way solution becomes probabilistic is if the IV is truly random
* When you recycle the IV it is not random anymore (IV = Counter)
* Patterns are not acceptable
* Relative to CBC it is less secure
* There is no order in Counter Mode
* CBC = Cipher Block Chain
* Integrity
  + It is a question of detection
  + A way of being able to verify if something bad happened or not
  + Two methods
    - Tag generation (MAC = Message Authentication Code)
    - Verification
      * Verify at the receiver side
      * Only the receiver should be able to verify
        + Hence, there needs to be some sort of key
        + Verification is just re-computing the mac & compares it with the tag.
  + CBC – MAC
    - AES can combine confidentiality and integrity
    - Good for application that are not time sensitive
* We are going to use Hash Function for the lab.
  + Shrink data into a small fixed size data relative to input size
  + Store this smaller data instead into a data structure
  + Hash Function is not one to one.
    - Multiple input matches to one output
    - This is called a Hash Collision
* Check Sum
  + Appended as part of the file
  + It verifies if what was downloaded is the original one
  + Nothing to do with security as much
* Hash Collision prevents Detection
* Confidentiality
  + It is a question of preventing
  + Methods
    - Encryption
    - Decryption
* Cryptographic Hash Function
  + One-way
    - You can compute the output base on the input
    - But not the other way around
    - Hardest for adversary to violate this
* Weak-collision resistance
  + Given x1
    - Should be computably infeasible to find x2 such that
    - H(x1) = H(x2)

1. Strong-collision resistance
   * Should be computably infeasible to find any pair of x1 and x2 where x1 is not equal to x2
   * Easier for adversary to violate this
2. If these are satisfied, then we have a fully secure hash function
3. There is no secure hash function now
4. Any hash function will expire
   * Collisions are bound to happen
5. MD5 is unsecure
6. MD5 is replaced with SHA0
7. SHA0 is replaced with SHA1
   * Many collisions happened here
8. SHA1
   * Used for checksum
   * Cause it is fast
9. SHA2 is not created by NSA
   * SHA256
     + The digest size is 256 bits
     + We are using this for the project
   * SHA512
     + The digest size is 512 bits
10. SHA3
    * Came out in 2015
    * This is secure
11. All SHA until SHA3 follow a structure by Merkel Damgaard
12. SHA3 uses something called “sponge”
13. Hash Functions are similar to AES

* H(k || m) can be compromised length extension attack
  + Vulnerable with length extension attack
* H(m || k)
  + Vulnerable with hash collision attack
* Hmac
  + H(k1||(h(k2||m))
  + This turned out to be a really good idea (double hashing)
* iPad – inner Padding
  + prepend to message
* oPad – outer Padding
  + prepend to hash
* tag = HMAC (k, m)
* SHA3 is faster than SHA2
* Hash is not parallizable or precomputable
* HMAC is faster than CBC Mac
* Integrity should be deterministic
* Confidentiality should be probabilistic
* The less you give the adversary the better
* Combine Confidentiality and Integrity
  + Encrypt message
  + Mac cipher text
  + Send the cipher text and the mac
* Combining HMAC and Encryption Makes Confidentiality Better
* Open SSL tags the message which isn’t good
* HMAC -> JWT
* JWT = Jason Web Token
  + Standard way of authentication
* HMAC used in KDF = Key Derivation Function
  + Have a key
    - Derive multiple keys from this key
    - Example
    - K
      * Want k1, k2, k3
      * Remember hmac(k, m)
      * M = 🡨------🡪
      * Next = m
      * For(I = 0; I < counter; i++)
        + Next = hmac (k, next)
      * Return next
      * The higher the counter the better from brute force attack
      * That is how you derive key from HMACs
    - PBKDF = password base key derivation function
      * K = h(password)
      * Then call the function