5 Hash functions

Questions & Answers

1. Hash functions create fixed length representations from arbitrary length inputs. When we usually call something a hash we are loosely referring to what academia is calling the "image". Study a basic hash function chart, its labels, all inputs, outputs and processes. See Crypto Intro and How to Break It.pdf, Slide 46
    - no answer needed here
2. What are hash functions good for. Think about five use cases or typical applications.
    - data structures: hash table -> https://www.tutorialspoint.com/data_structures_algorithms/hash_data_structure.htm (not really related to crypto)
    - key generation: number whose entrophy is too low (not random enough) might be hashed
    - password verificaton (authentication) -> compare previously hashed pw with hash of entered pw (to avoid having to store pw in cleartext)
    - data ingegrity checks
    - authentication via signature (signed hash): for example to authenticate servers (in combination public key crypto)
    - licence checks (also via signature)
3. With hash functions come three important properties: collision resistance, pre-image resistance and second pre-image resistance. Describe the three based on Crypto Intro and How to Break It.pdf, Slides 49 to 51 and give hints on which is the easiest to exploit.
    - collision resistance
        - When two different inputs result in the same hash (image) we have a collision. When it is difficult to deliberatly create 2 different inputs resulting in the same hash, the hash algorithm has a good collision resistance.
    - pre-image resistance
        - It is difficult to find any pre-image (data before hashing) that results in a target hash/image. Basically we know the hash result we want, but we search the data resulting in it.
    - second pre-image resistance
        - Similar to pre-image resistance but here we know the pre-image and the image/hash. Now we try to find another pre-image, that is different from the one we already have, but still results in the same hash. Example: We want to forge a document that is signed. We now changed some parts of the text, and add spacess until we get the same hash again.
4. Assuming we have a perfect hash function that produces 128-bit output (Same size as with MD5). How resistant is the said function against collisions. Can you calculate the computational effort and would this be sufficient protection nowadays? See Crypto Intro and How to Break It.pdf, Slide 49

- Not sure i got it right but i would assume on average you have to try 2^64 times till you get a collision. Reason: the output of the "perfect" hash functions should appear to be completly random and there can be 2^128 different hashes (given its size). So with 2^128 possibilities, one has to try 2^64 times on average.

5. If we need a secure hash function to assure proper integrity protection for the next decade. What would be minimal size for a good function? Check https://www.keylength.com/ and argue your assumptions and recommendation.

    - I would recommend 256bit or higher as this is what is mainly recommended by the keylength tool
    - Some considerations:
        - Machines are becoming faster and faster but moore's law is not a law of nature and at some point progress in performance will slow down
        - quantum computers will probably andvance and some crypto algorithms might get broken. However, according to wikipedia SHA-3 is at least to some degree (and to my understanding) resistant to quantum computer attacks. Therefore, I assume 256bit is still ok.