One-way Functions

general idea

functions that are easier to compute in one direction

but are very difficult to compute in the opposite direction

hash(X) -> Y

given Y, if finding X is difficult, then this function is a one-way function

trap-door one-way function

a special type of a one-way function where computing the opposite direction is easy

but only if the additional "trapdoor" information is known

hash(X + Y) = Z

given Z and some additional information, it may be possible to compute X or Y or both

one-way hash function

takes a variable-length input string (called a pre-image)

and transforms it into a fixed-length output string (called the hash value – or digest)

MD5, SHA-1, SHA-256, etc are common examples of one-way hash functions

also referred to in different ways in cryptography:

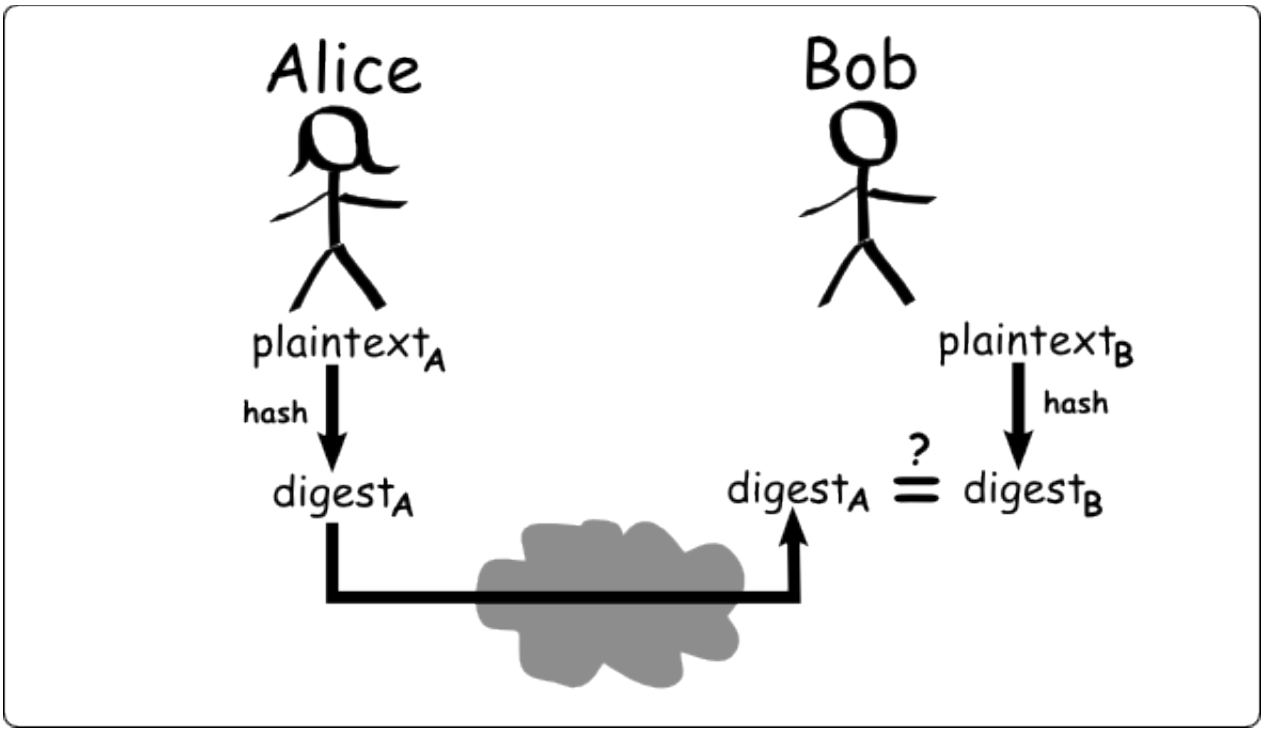
message digest

cryptographic checksum

message integrity check

manipulation detection code

and so on...



properties of an ideal hashing algorithm

the same input will always result in the same output

H(a) = b *H(a) = b*

regardless of the size of an input, the output length will always be the same

length(H(abc)) = 32 *length(H(abcdefg)) = 32*

it is infeasible to get the original pre-image back from the hash value

H(a) = b *if only b is known, a remains unknown*

even a small change in the message will result in a very different output

H(abcdefghij) = d *H(abcdefghi) = x*

it is infeasible to find two messages with the same hash value

this is known as collision resistance

H(abced) = b *H(cdefgh) = b (this must not occur!)*

hash collisions

when two different messages happen to produce the same output

of course, when processed through the same hashing algorithm

collisions have been observed in MD5 and SHA-1

they have been effectively rendered obsolete in security applications

but they can still be used in checksum applications, for example

message authentication code (MAC)

a piece of information that is used to authenticate a message (its origin)

and verify that it has not been altered (its integrity)

the receiver/verifier uses a secret key to authenticate the message

basic algorithm:

select a random secret key from the keyspace

share the secret key between the receiver and the sender

generate a tag (hash value) from the message and the key

append the tag to the message

receiver generates the tag by using the received message and the shared secret key

if this tag matches the tag received with the message, then the message is authenticated

HMAC: an interesting (and standard) MAC

specifically, HMAC-SHA-256 (because it internally uses SHA-256)

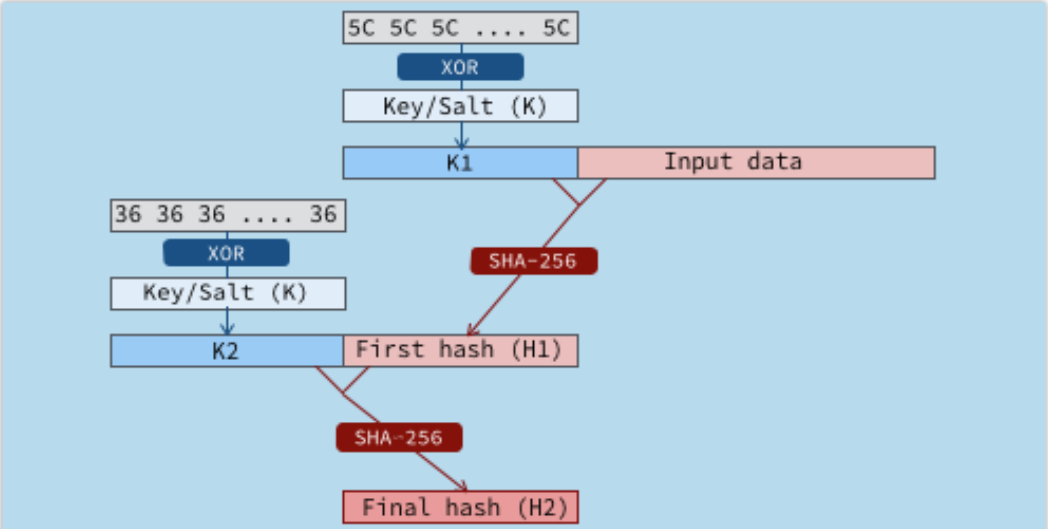
a special way of using SHA-256 without just having a straight hash

take a random key or salt K, flip some bits (XOR with 5C5C5C…) → K1

compute SHA-256 of K1 plus the user's password → H1

flip a different set of bits in K (XOR with 363636…) → K2

compute SHA-256 of K2 plus H1 → H2



HMAC performs two iterations of a chosen cryptographic hash to create a "keyed hash"

sort of a hash of a hash of the input data

with the added randomness of XORing the key with two different “pads”

known as inner and outer pads