Hash Functions

Introduction
Random Numbers
Symmetric Encryption
Hashes
Asymmetric Encryption
Certificates
Signatures
SSL/TLS
SSH
VPN
Email
Disk Encryption
Attacks

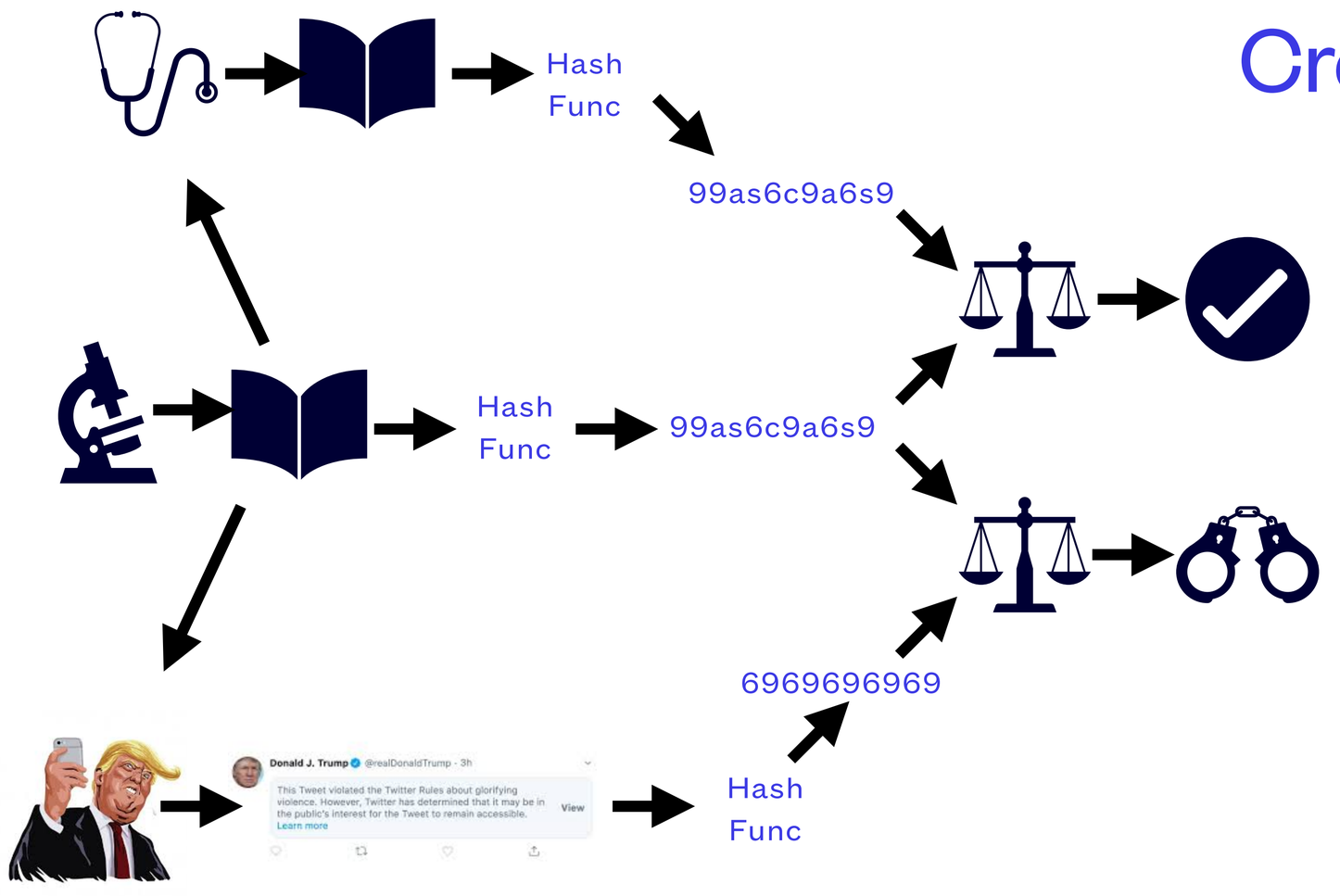
Applied Cryptography

The Four Primitives

- There are four primitives which are considered the building blocks of digital cryptography
  - Random Number Generation
  - Symmetric Encryption
  - Asymmetric Encryption
  - **Hash Functions**
- These primitives get combined to achieve the CIA (confidentiality, integrity, authentication) goals of cryptography

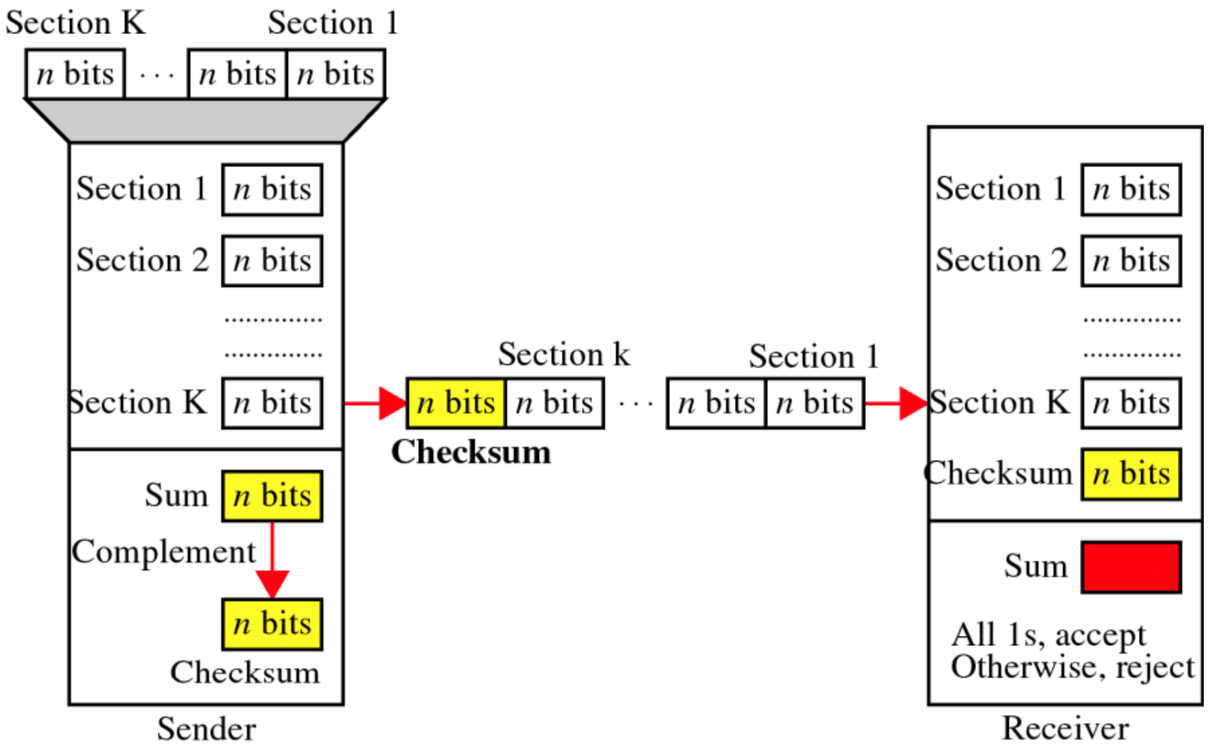
Hash Functions

Creating a digest or hash



- Hash functions are mathematical functions that take source data in any quantity and use it to generate a block of data of a fixed size called a digest or informally, a hash
- The digest value changes significantly if the source data changes even a small amount, allowing us to use the digest to easily recognize if the data has been altered since the hash was created
- The digest can be viewed as a trustworthy identifier for the data
- The digest block size is dependent on the algorithm, not the input size

Hash Functions
Irreversible

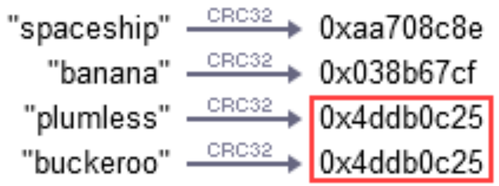


https://www.faceprep.in/computer-networks/computer-networks-error-detection/

- Hash functions are one-way functions
  - The data is generated in such a way as to render it impossible to reverse the process
  - Digests are not secret because they reveal nothing about the data they were generated from
- The simplest and oldest of these is a checksum
  - Obtained by simply adding all the pieces of the data together and expressing the result with a fixed size number
  - Still used for error checking in protocols, not reliable because it only detects single bit errors or changes
  - CRC was the successor and could detect larger changes, but still not very useful today
  - Many digest generating and checking programs include the word "sum" in their names

Hash Functions
Collisions

Number of **32-bit** hash values	Number of **64-bit** hash values	Number of **160-bit** hash values	Odds of a hash collision	
77163	5.06 billion	1.42×10^{24}	1 in 2	
30084	1.97 billion	5.55×10^{23}	1 in 10	
9292	609 million	1.71×10^{23}	1 in 100	Odds of a full house in poker 1 in 693
2932	192 million	5.41×10^{22}	1 in 1000	Odds of four-of-a-kind in poker 1 in 4164
927	60.7 million	1.71×10^{22}	1 in 10000	
294	19.2 million	5.41×10^{21}	1 in 100000	Odds of being struck by lightning 1 in 576000
93	6.07 million	1.71×10^{21}	1 in a million	Odds of winning a 6/49 lottery 1 in 13.9 million
30	1.92 million	5.41×10^{20}	1 in 10 million	
10	607401	1.71×10^{20}	1 in 100 million	Odds of dying in a shark attack 1 in 300 million
	192077	5.41×10^{19}	1 in a billion	
	60740	1.71×10^{19}	1 in 10 billion	
	19208	5.41×10^{18}	1 in 100 billion	
	6074	1.71×10^{18}	1 in a trillion	
	1921	5.41×10^{17}	1 in 10 trillion	Odds of a meteor landing on your house 1 in 182 trillion
	608	1.71×10^{17}	1 in 100 trillion	
	193	5.41×10^{16}	1 in 10^{15}	
	61	1.71×10^{16}	1 in 10^{16}	
	20	5.41×10^{15}	1 in 10^{17}	
	7	1.71×10^{15}	1 in 10^{18}	



https://preshing.com/20110504/hash-collision-probabilities/

- Hash collisions occur when 2 source data blocks with different content happen to produce the same hash function output
- Hash functions are considered cryptographically insecure if they produce collisions at an unacceptable rate or collisions can be generated on purpose, so these hash functions must be collision resistant
- md5 and SHA-1 are deprecated as crypto hash functions because they can be made to produce collisions at will

Hash Uses

- A digest of plaintext can be used as
  - An identifier for a password that can be used to check if a given password matches
  - Key or passphrase verification for encrypted data to avoid wasting time decrypting
  - Timestamp guarantor
  - Certificate integrity check
  - Integrity verification for arbitrary data
- Any situation that requires a way to check if input matches previously seen data
- A digest mismatch does not tell you how the tampering or corruption occurred, only that it happened

Integrity Verification

- Symmetric encryption algorithms apply a mathematical formula to a key and some ciphertext to produce plaintext
- Many algorithms will not fail if the ciphertext isn't encrypted data, so if the data you are decrypting isn't actual ciphertext that was generated by the same cipher and a matching encryption key, then the decryption algorithm can still be applied but the result will be garbage
- Digests stored with the encrypted data allow a decryption program to verify if what is in the the ciphertext is valid, by hashing the decrypted data and checking if it matches the digest stored in the ciphertext
  - If it doesn't, the key was wrong or the data has been altered
- Some algorithms incorporate this to enable integrity checks on the data, the key, or both
- It is up to individual software tools to decide whether to use digests, and how
- These are called AE or AEAD ciphers

- **AE**
  - Aunthenticated Encryption
- **AEAD**
  - Authenticated Encryption with Associated Data

MD Series of Hash Functions

- A series of hash functions created by Ron Rivest
- MD2 and MD4 are compromised and should never be used
- MD5 collision rate is unacceptable, deprecated since 2017 but still used in too many places - md5sum command and other common programs keep noobs using it
- MD6 was produced around 2008 to compete in the NIST SHA3 competition
- It did not win, speed and lack of proof of security were the main reasons

- **MD**
  - Message Digest
- **NIST**
  - U.S.A. National Institute of Standards and Technology

SHA Series of Hash Functions - SHA-0

- A family of hash functions
- Driven by and incorporated in the DSA, and its replacements
- Published by NIST as a FIPS
- SHA-0 was a renaming of the original SHA
  - A hash function which produces a 160-bit digest
  - Security level of 80 bits
    - Brute force would require hashing 2^{80} messages to find a collision
  - Designed by the NSA and published as SHS in 1983
  - Replaced by SHA-1 in 1995 and not approved for use since then

- **SHA**
  - Secure Hash Algorithm
  - Created by the NSA
- **FIPS**
  - U.S.A. Federal Information Processing Standard
- **SHS**
  - Secure Hash Standard
  - FIPS PUB 180
- **NSA**
  - U.S.A. National Security Agency
- **DSA**
  - Digital Signature Algorithm

SHA Series of Hash Functions - SHA-1

- SHA-1 is a 160-bit hash function
  - designed by the NSA and published in 1995 as FIPS PUB 180-1
  - improved on SHA-0 in a trivial way to eliminate a significant flaw in SHA-0
  - Very similar in design to MD5
  - Broken by a trio of Chinese cryptographers in 2005, meaning collisions made to occur without using brute force
  - Approval for its use by federal agencies withdrawn since 2010 for purposes other than verifying old signatures and timestamps
  - Still widely used as a base for many other functions and protocols, primarily as an identifier instead of for security
  - e.g. github uses it for consistency checks

SHA Series of Hash Functions - SHA-2

- SHA-2 replaced SHA-1 with 6 hash functions derived from the SHA-1 functions
  - designed by the NSA
  - significantly different and improved over SHA-1
  - SHA-256 and SHA-512 are the novel functions
  - Block sizes are 32-bits and 64-bits
  - SHA-256, SHA-512, SHA-384 published as FIPS PUB 180-2 in 2001
  - SHA-224 added in 2008 as part of FIPS PUB 180-3, became the minimum security level (112) recommendation from NIST in 2011, and requirement in 2014
  - SHA-512/224 and SHA-512/256 added in 2012 as part of FIPS PUB 180-4
  - SHA-2 is currently in wide use

SHA Series of Hash Functions - SHA-3

- SHA-3 does not replace SHA-2
- Completely new functions not derived from SHA-2
  - Chosen by competition in 2012
  - Much larger internal state than SHA-2 variants, yet faster to use
  - Makes brute force attack wildly impractical due to both time and memory usage
  - SHA-3 broadens the NIST hash function toolkit
  - SHA-3 is part of the Keccak family of crypto primitives and the hash function has spawned many derivative works
  - SHA-3 wikipedia page is pretty good

Hashing Commands

- md5sum, sha1sum, sha256sum, sha512sum, etc.
- b2sum for Blake hash
- tthsum for Tiger hash
- openssl for uncommon hashes
- certutil for windows, only does specific hashes
- many commands allow for generation of hashes while performing other duties
  - passwd
  - htpasswd
  - dc3dd