University of Basel

Center for Innovative Finance

Bitcoin, Blockchain and Cryptoassets

Hash Functions

Prof. Dr. Fabian Schär
University of Basel

Release Ver.: (Local Release)
Version Hash: (None)
Version Date: (None)

What Is a Hash Function?

Deterministic algorithm (function, $H()$) that maps data of quasi-arbitrary size (pre-image, m) to fixed-length bit string (hash value, h).

$$h = H(m) \tag{1}$$

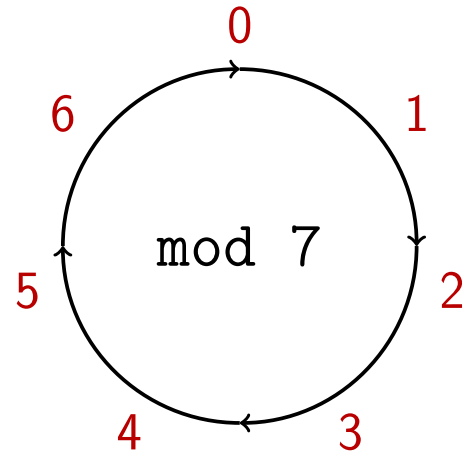
Application fields: (non-exhaustive)

- Data protection
- Verification and authentication
- Proof-of-work
- Data lookup optimization
- Error detection

Simplified Hash Function Using Modular Arithmetic

Modular arithmetic

$$8 = 1 \ (mod\ 7)$$
$$2 + 3 = 5 \ (mod\ 7)$$
$$2^5 = 2 \ (mod\ 5)$$



$\Rightarrow$ Useful to create one-way functions.

Simplified hash function

$$h \ (mod\ 12) = m \tag{2}$$

This example accepts any numeric pre-image m and returns a corresponding hash value $h \in \{0, ..., 11\}$.

Simple Hash Functions: Checksums

Checksums find many applications to detect human errors in data entry or faulty / incomplete data transmission.

IBAN Example:

Country Code		Bank Code	Account No.
CH	56	0483501	2345678009
	Checksum	BBAN	

Steps:

1. Take BBAN and append Country Code and 00 (empty checksum).
2. Replace any letters with number: 9 + `<position in alphabet>`.
3. Perform modulo 97 operation.

   $\rightarrow$ 4,835,012,345,678,009,121,700 $= 42$ (mod 97).
4. and subtract the result from 98.

   $\rightarrow$ $98 - 42 = 56$.

Further Examples and Limitations

Other checksum examples:

- Credit Cards
- Vehicle Identification Numbers (U.S. and Canada)
- Radio protocols (often modulo on bytes)
- Communication (parity bits)

Purpose to **detect accidents**, not to **prevent attacks**:

CH5604835012345678009 vs. CH5604835012345687709

$\Rightarrow$ For security purposes, simple hashes are of limited use.

Cryptographic Hash Functions

Additional criteria:

1. Approximately uniform hash value distribution.
2. Quick to compute for any given pre-image.
3. Trap-door: Infeasible to generate pre-image from hash value.
4. Avalanche effect: Small change in input results in totally different output.
5. Very low collision probability: Unlikely that two pre-images generate the same hash value.

In Bitcoin context, the functions `SHA2.256` (in short `SHA256`) and `RIPEMD160` are used. Both satisfy the above criteria.

Avalanche Effect with SHA2

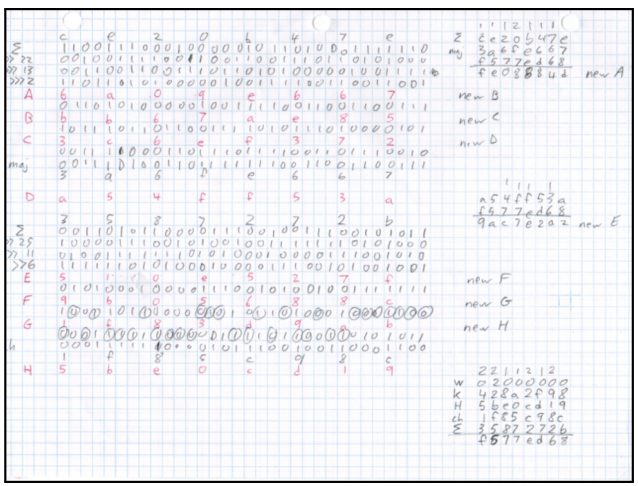
SHA256(<*pre-image*>)

This is the pre-image

⇒ 5e13fffedd642aaddea7872463fd44d8b8a336cb822ebd4c12d9e6282c88cea8

this is the pre-image

⇒ aba56d7c281f8dacb1076baed182538d2ce21494075f7377982bf9d5d08e6ccf



- Nonlinearity due to choice, majority, mod, rotation and shifting operations.
- Efficient for computers vs. 0.67 hashes / day by hand.
- Video: Hash value by hand.

Low Collision Probability

Due to the fixed size of h, the corresponding hash function $H()$ can only produce a finite set of distinct hashes.

`SHA256`:

- All possible combinations of 256 Bits, i.e., 2^{256} h.
- In base 10, the number of combinations corresponds to 115,792,089,237,316,195,423,570,985,008,687,907,853,269, 984,665,640,564,039,457,584,007,913,129,639,936

Probability of a hexadecimal hash with certain characteristics:

$$P(h \leq 0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF) = \frac{1}{16}$$

$$P(h \leq 00FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF) = \left(\frac{1}{16}\right)^2$$

Script Example: Brute Forcing Attempt

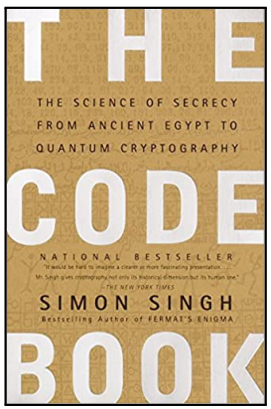
Goal: Find a nonce that, together with a fixed input string, returns a `SHA256` hash with, preset characteristics, in this case x leading zeros.

Script logic:

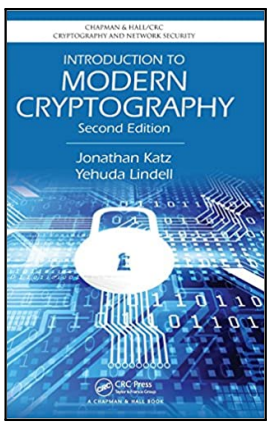
1. Take `input` and add `nonce`, starting at 0.
2. Compute `SHA256` hash with `<input>⌒<nonce>` as pre-image.
3. Compare resulting `hash` against minimum zeros:
   - Criteria not met: Increase nonce by 1 and return to step 2.
   - Criteria met: Return `hash`, `nonce`, and time it took to find it.

↗ Example script in Python.

References and Recommended Reading



The Code Book
Simon Singh
ISBN: 978-0385495325



Introduction to Modern Cryptography
Jonathan Katz and Yehuda Lindell
ISBN: 978-1466570269