

CSE446: Blockchain & Cryptocurrencies

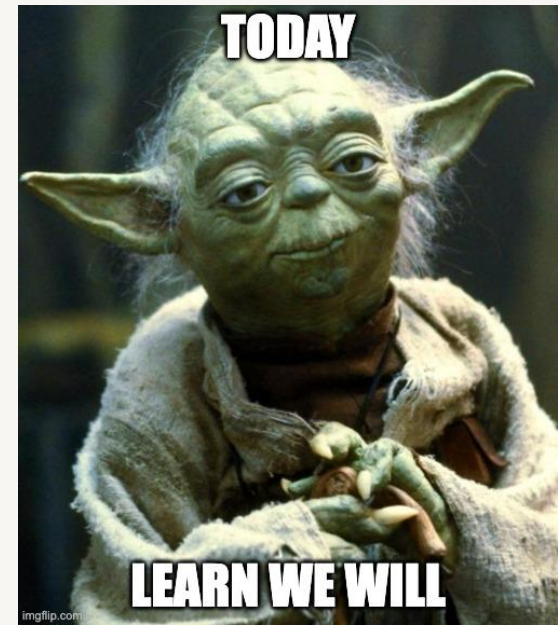
Lecture – 2: Cryptography Review



Inspiring Excellence

Agenda

- Cryptography review
 - Cryptographic hash functions
 - Hash pointer
 - Merkle Tree
 - Symmetric encryption
 - Asymmetric encryption (Public-key encryption)
 - Digital signature



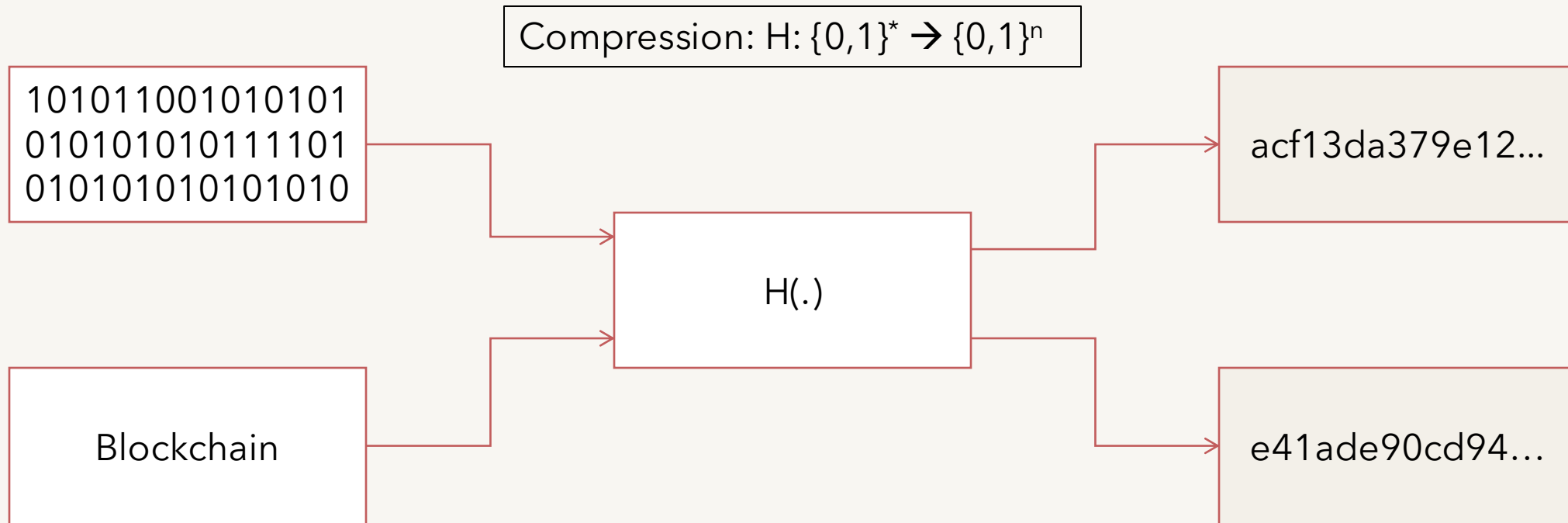
This lecture has been prepared from multiple sources:

- Textbook
- <https://github.com/PratyushRT/blockchainsS21/wiki>
- <https://github.com/sebischair/bbse>

Cryptographic hash function

- A hash function is a mathematical function with the following three properties
 - Its input can be any string of any size
 - It produces a fixed size output
 - It is efficiently computable
 - computing the hash of an n -bit string should have a running time of $O(n)$

Cryptographic hash function



Cryptographic hash function

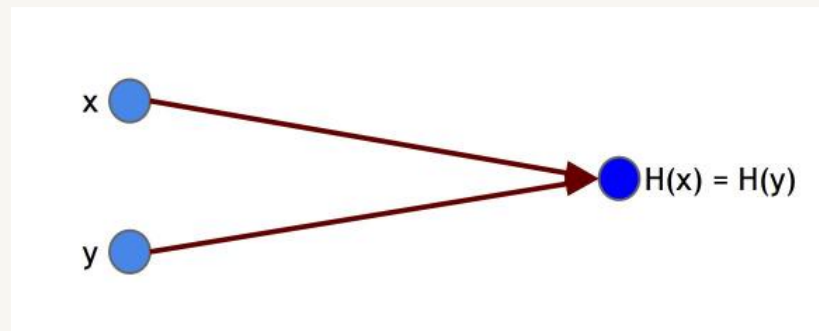
- A cryptographic hash function is a general hash function that should satisfy these properties
 - collision-resistance
 - preimage resistance
 - hiding
 - puzzle-friendliness

Cryptographic hash function

- A cryptographic hash function is a general hash function that should satisfy these three properties
 - collision-resistance
 - preimage resistance
 - hiding
 - puzzle-friendliness
- Must-have
- Desirable for certain blockchain systems

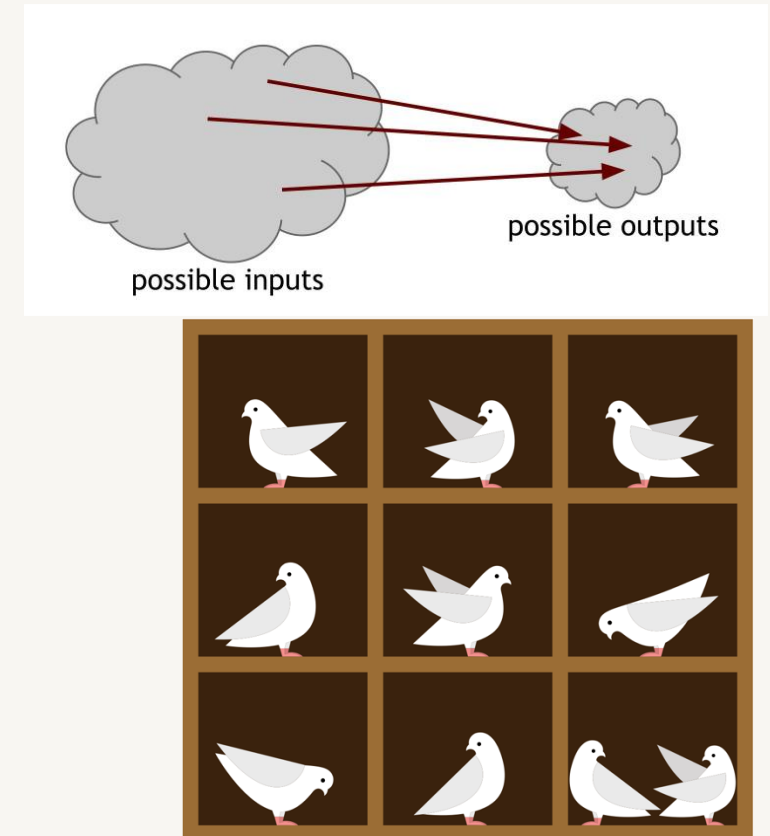
Cryptographic hash function: collision resistance

- A hash function is said to be collision resistant if it is infeasible to find two values, x and y , such that $x \neq y$, yet $H(x) = H(y)$



Cryptographic hash function: collision resistance

- A hash function is said to be collision resistant if it is **infeasible** to find two values, x and y , such that $x \neq y$, yet $H(x) = H(y)$
- Infeasible \rightarrow hard to find a collision, but not, no collisions exist
- The input space is $>$ the output space (the input space is infinite, while the output space is finite)
 - there must be input strings that map to the same output string (the pigeonhole principle)
- But it will be hard to find these



Cryptographic hash function: collision resistance

- How to find a collision?
- Choose $2^{256} + 1$ distinct Input for a hash function with 256 bit output
- Calculate hash for each input and check if the output matches with any previous hash
- Since input size $>$ output size, there must be a match (collision)
- Try 2^{130} randomly chosen inputs, 99.8% chance that two of them will collide
 - Examining roughly the square root of the number of possible outputs (the birthday paradox)
 - The birthday paradox is that, counterintuitively, the probability of a shared birthday exceeds 50% in a group of only 23 people

Cryptographic hash function: collision resistance

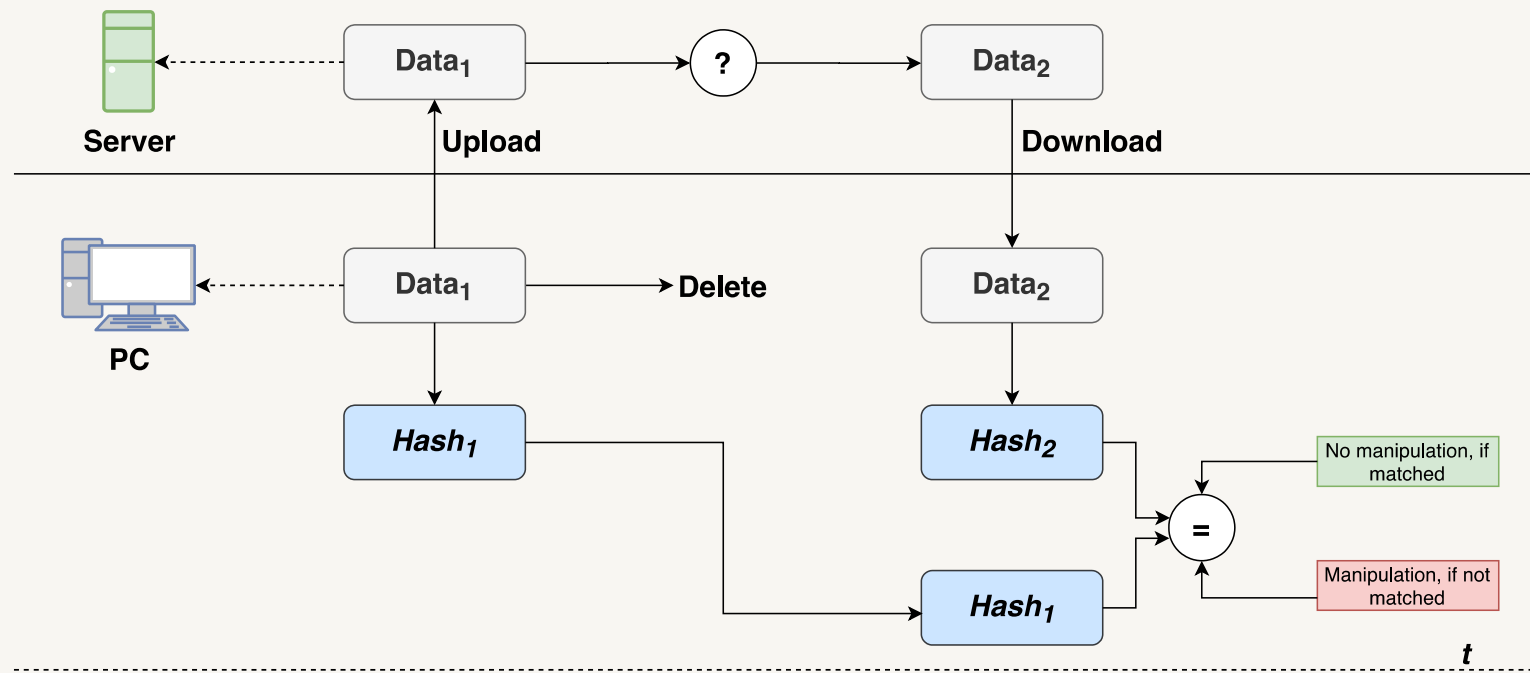
- Is finding collision computationally feasible?
- A 256-bit hash function
 - worst case: $2^{256} + 1$ times
 - best case: 2^{128} times on average
- If a computer calculates 10,000 hashes per second, 10^{27} years to generate 2^{128} hashes!

Cryptographic hash function: collision resistance

- The previous way was a brute-force method
- Is there any other optimised method available for finding collisions?
- Yes, for some hash functions: $H(x) = x \bmod 2^{256}$
 - Generates a 256 bit output and easily computable
 - But returns the last 256 bits of the input. One collision: 3 and $3 + 2^{256}$
- For others (e.g. SHA-256), we don't know yet

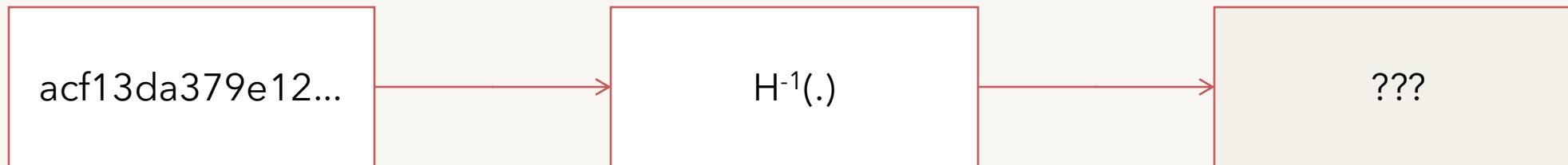
Collision resistance: application

- Message digest: a hash of any input, e.g. bits, random strings, characters or even files



Cryptographic hash function: pre-image resistance

- H is a hash function
- For essentially all pre-specified outputs y , it is computationally infeasible to find an x such that $H(x) = y$
- H is also called a one-way function



Cryptographic hash function: pre-image resistance

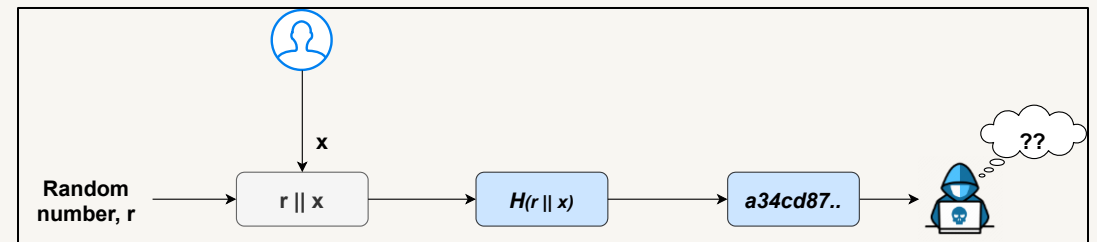
- If x is drawn from a uniform distribution with a large number of elements, then inverting $H(x)$ is hard
- But what if x is drawn from a low min-entropy distribution?
 - In information-theory, min-entropy is a measure of how predictable an outcome is
 - High min-entropy captures the intuitive idea that the distribution (i.e., random variable) is very spread out

Cryptographic hash function: pre-image resistance

- But what if x is drawn from a low min-entropy distribution?
- Let the sample space is $X = \{h, t\}$
 - $H(x) = y$
- Can an attacker find the value of x given y ?

Cryptographic hash function: hiding

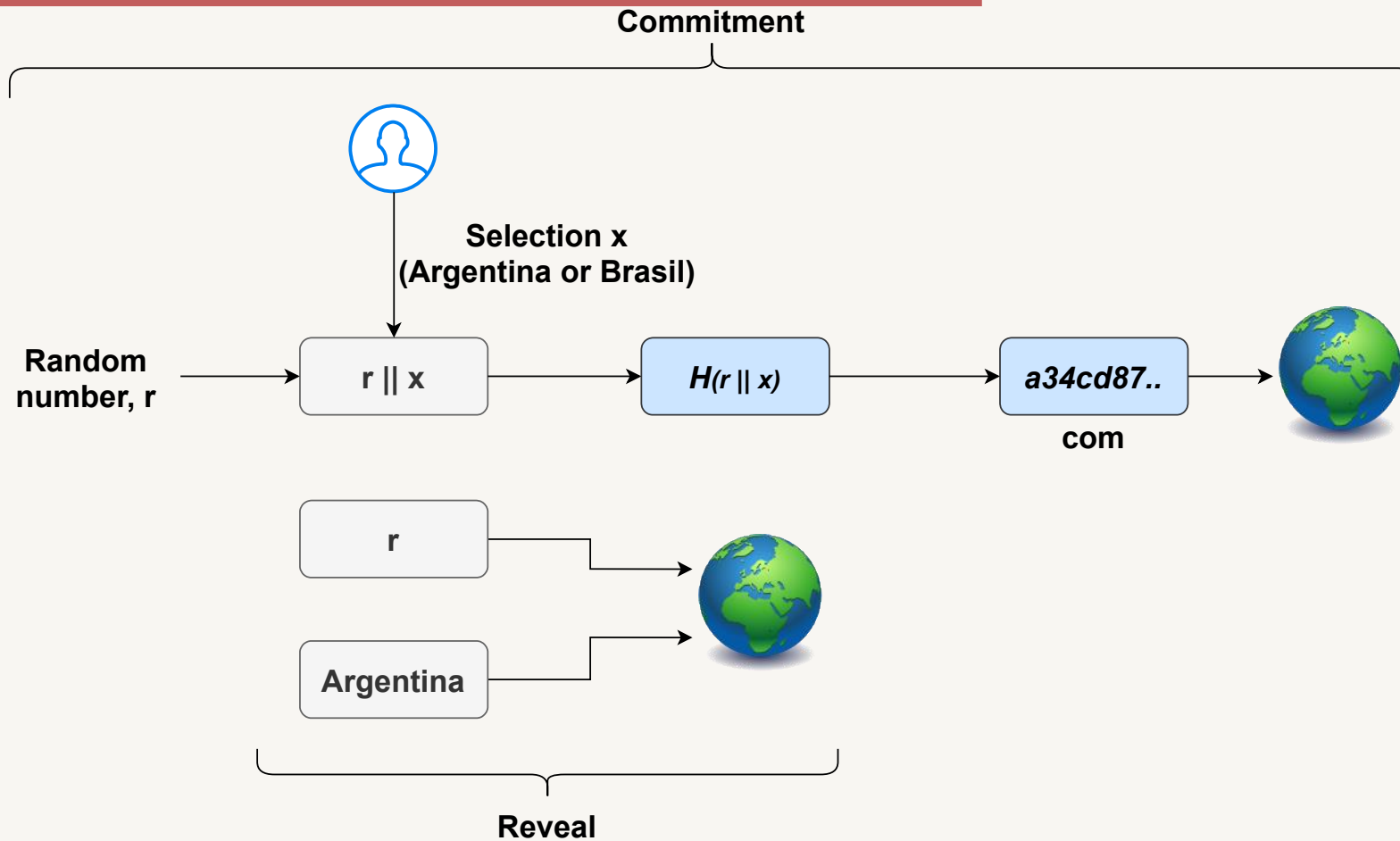
- A desirable property for a cryptographic hash function is hiding which tackles x picking up from a low min-entropy distribution
- A hash function H is hiding if
 - when a secret value r is chosen from a probability distribution that has high min-entropy,
 - then given $H(r \parallel x)$ it is infeasible to find x



Hiding: application

- Commitment scheme
- Want to “seal a value (who will win 2026 world cup?) in an envelope”, and publish it
 - Commit to a value (Argentina 😊) -> this is commitment
- Reveal your commitment to anyone -> open the envelope and verify your commitment

Hiding: commitment scheme



Hiding: commitment scheme

- $\text{com} := \text{commit}(\text{msg}, \text{key})$
 - msg is the message and key is the random number used once
 - commit is essentially a hash function operating over the concatenation of msg and key
- $\text{verification} := \text{verify}(\text{com}, \text{msg}, \text{key})$
 - Checks and returns whether msg and key produce the same result as com
- Security properties:
 - Hiding: Given com, no adversary can find msg
 - Binding: No adversary can find $(\text{msg}, \text{key}) \neq (\text{msg}', \text{key}')$ such that $\text{verify}(\text{commit}(\text{msg}, \text{key}), \text{key}', \text{msg}') == \text{true}$

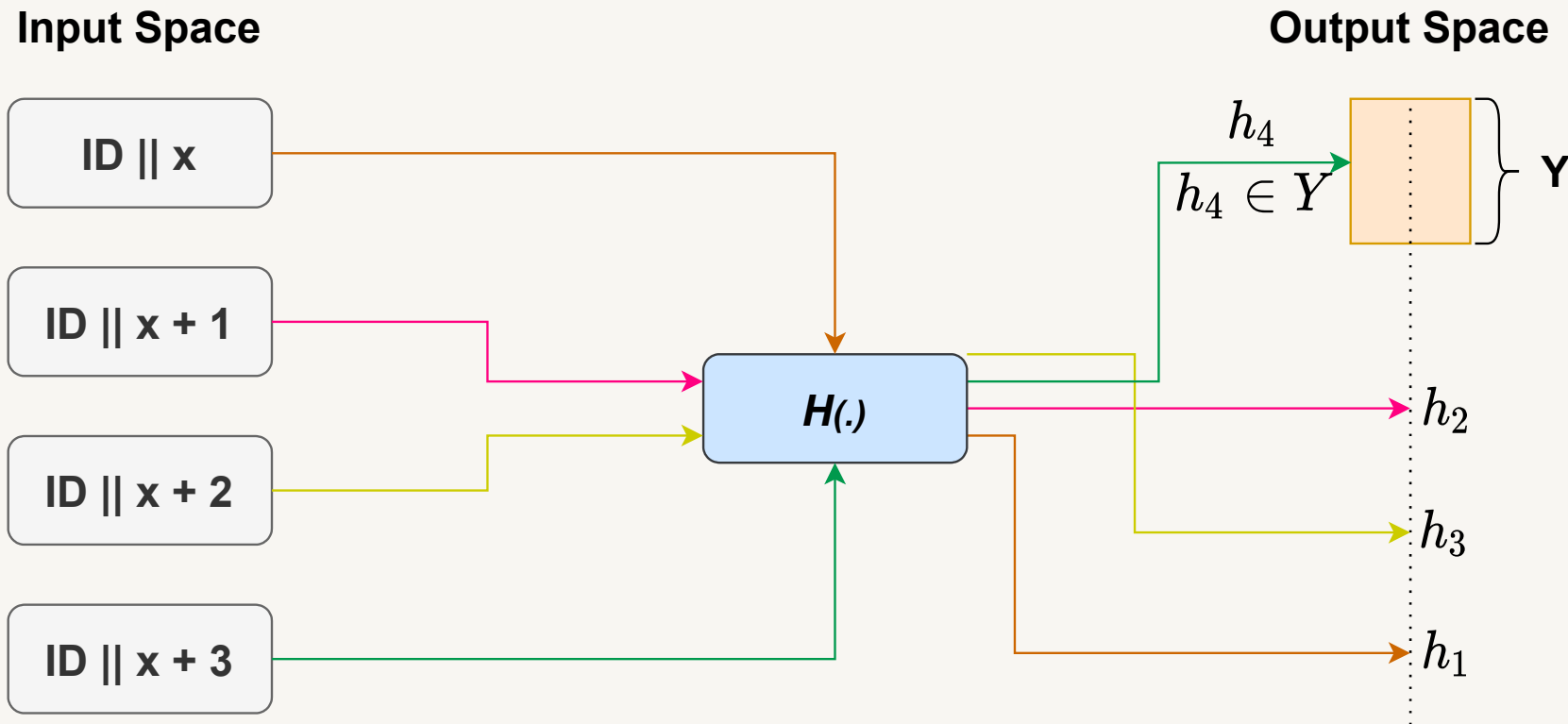
Cryptographic hash function: puzzle-friendliness

- A hash function H is said to be puzzle-friendly if
 - for every possible n -bit output value y , if k is chosen from a distribution with high min-entropy,
 - then it is infeasible to find x such that $H(k \parallel x) = y$ in time significantly less than 2^n
- If a hash function is puzzle friendly, then there is no solving strategy for this type of puzzle that is much better than trying random values of x

Puzzle~friendliness: application

- Search puzzle
- Consists out of:
 - A hash function H : Computes the *puzzle results*
 - A value id : *puzzle-ID* (makes solutions to the puzzle unique, should not be known in advance, otherwise pre-computation is possible)
 - A target set Y , for a valid solution z , $z \in Y$
 - Computation: $z = H(\text{puzzle-ID} \parallel x)$
 - x changes until $z \in Y$

Puzzle~friendliness: application



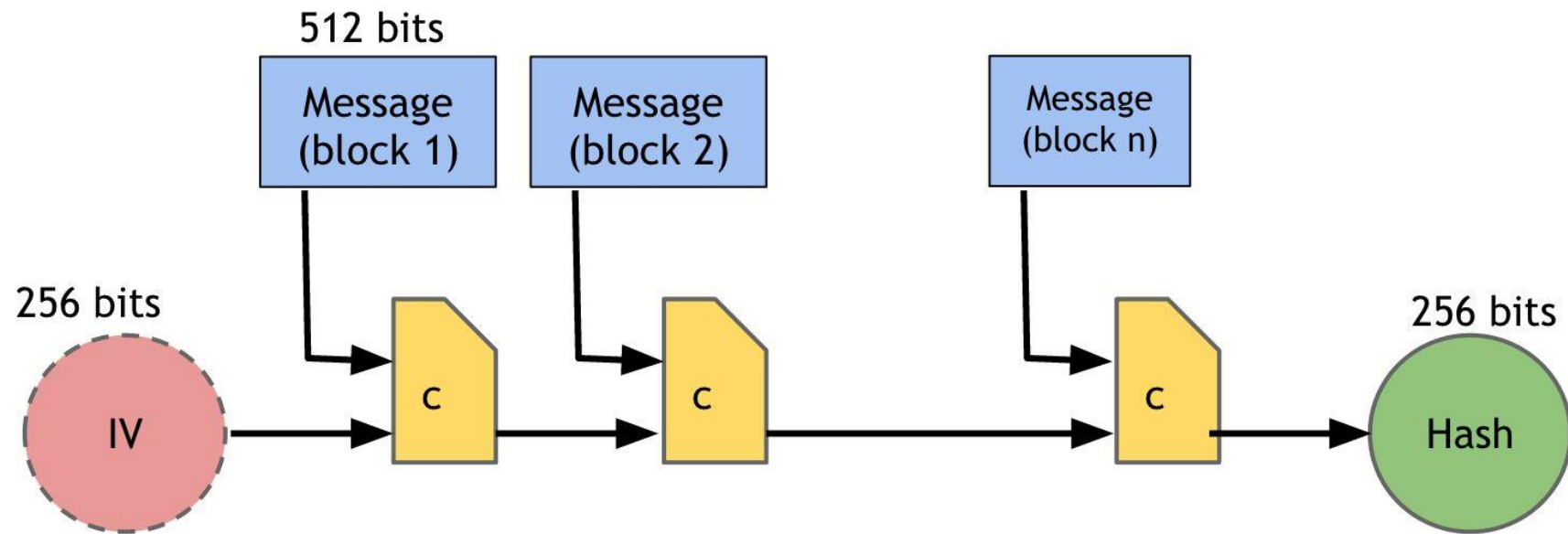
Family of hash functions

- MD5 (Message Digest 5) -> Currently considered broken!
- Secure Hashing Algorithm 1 (SHA-1) -> Currently considered broken!
- Secure Hashing Algorithm 2/3 (SHA-2/3) -> safe to use, SHA-3 preferable

SHA-256 (SHA-2)

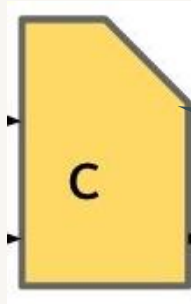
Break the message in the multiples of 512 bits, pad 0s in the last block to make it a 512 bit block

Merkle-Damgard
Construction

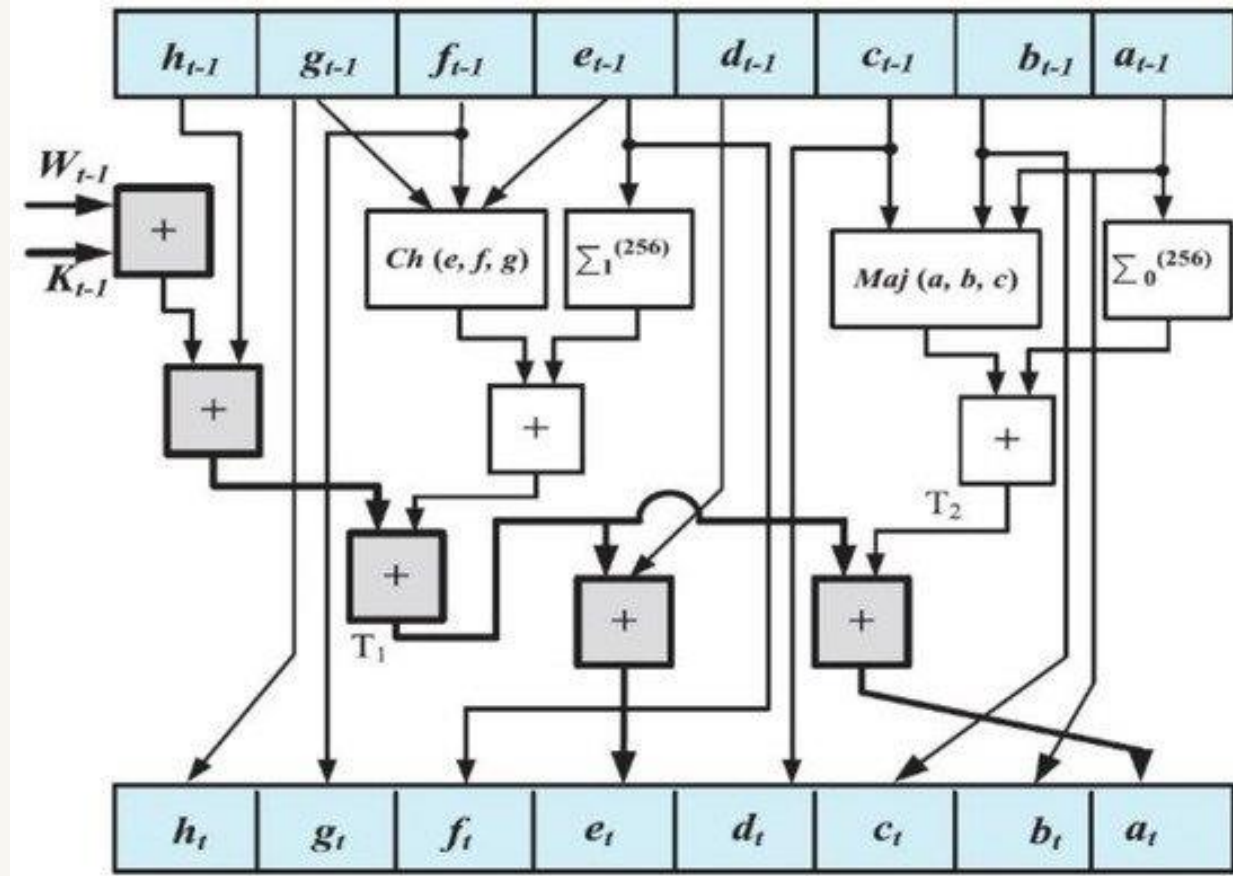


Theorem [Merkle-Damgard]: If c is collision-resistant, then SHA-256 is collision-resistant

SHA~256 (SHA~2)



SHA-256 (SHA-2)



https://www.mdpi.com/entropy/entropy-21-00577/article_deploy/html/images/entropy-21-00577-g001-550.jpg

