CS 394B Introduction

Marco Canini

Cryptography Hash Functions I

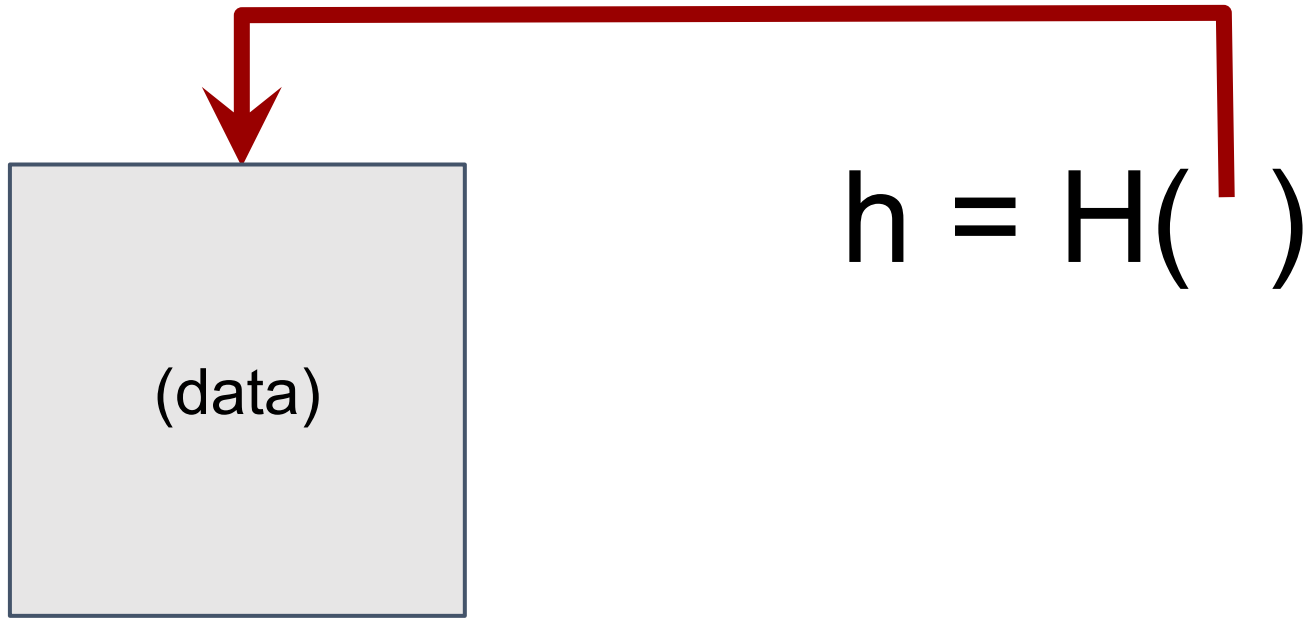
- Take message *m* of arbitrary length and produces fixed-size (short) number *H(m)*
- One-way function
  - Efficient: Easy to compute *H(m)*
  - **Hiding property:** Hard to find an *m*, given *H(m)*
    - Assumes “m” has sufficient entropy, not just {“heads”, “tails”}
  - **Random:** Often assumes for output to “look” random

Cryptography Hash Functions II

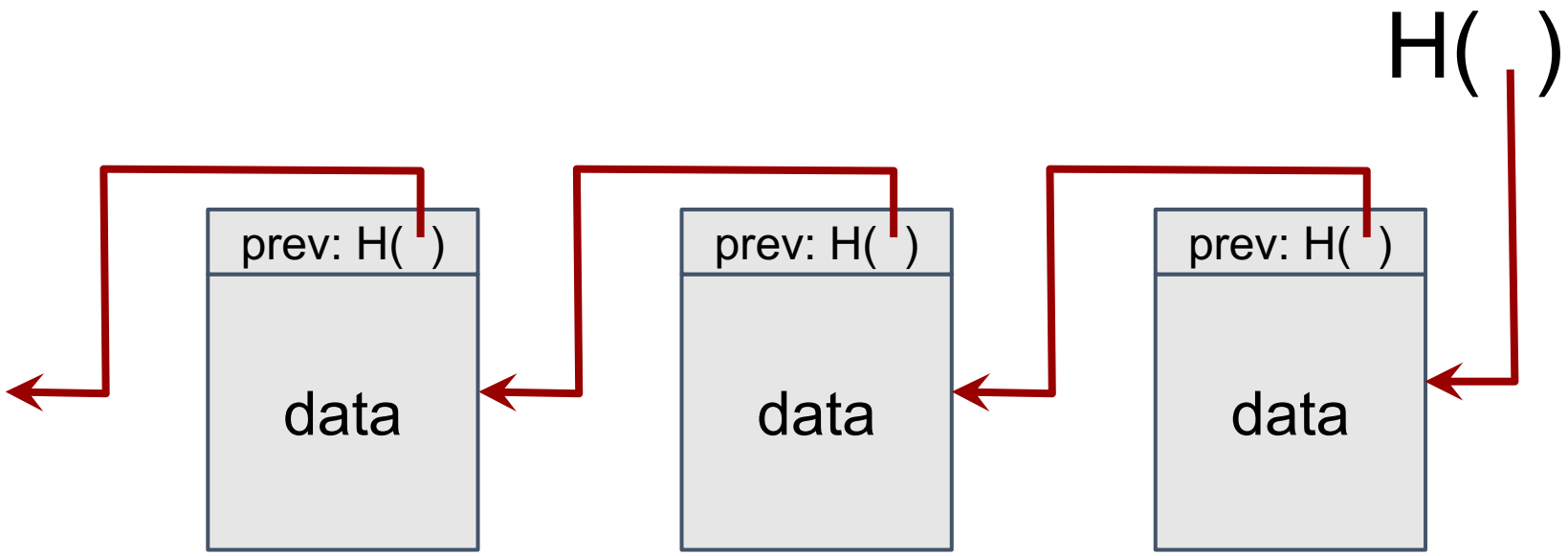
- Collisions exist: | possible inputs | >> | possible outputs |
  … but hard to find

- Collision resistance:
  - Strong resistance: Find any m != m' such that H(m) == H(m')
  - Weak resistance: Given m, find m' such that H(m) == H(m')
  - For 160-bit hash (SHA-1)
    - Finding any collision is birthday paradox: 2^{160/2} = 2^80
    - Finding specific collision requires 2^160

Hash Pointers

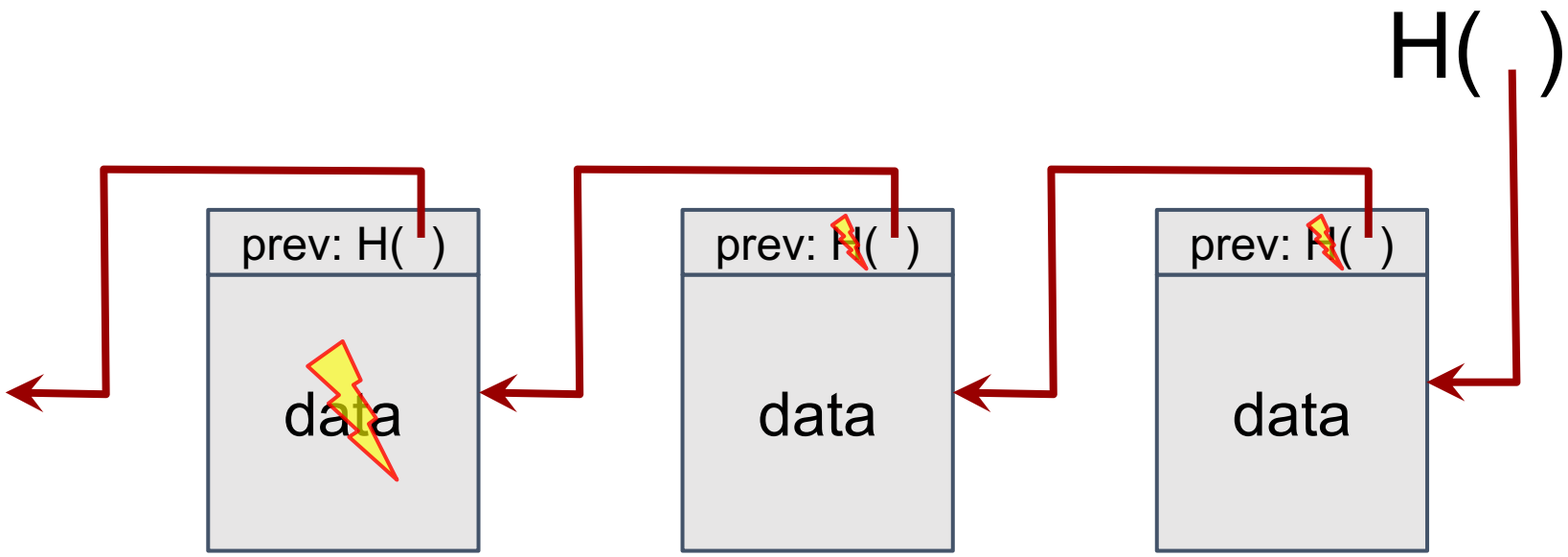


Hash chains



Creates a “tamper-evident” log of data

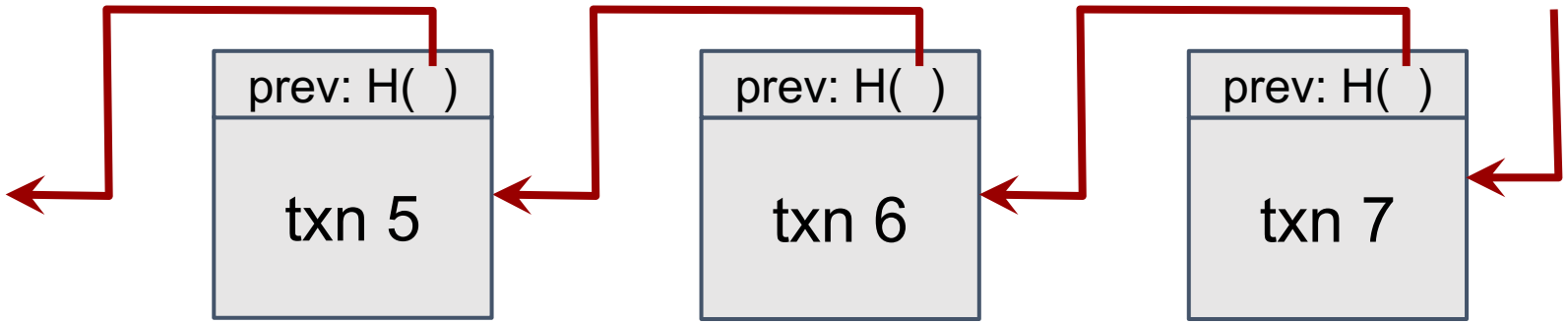
Hash chains



If data changes, all subsequent hash pointers change

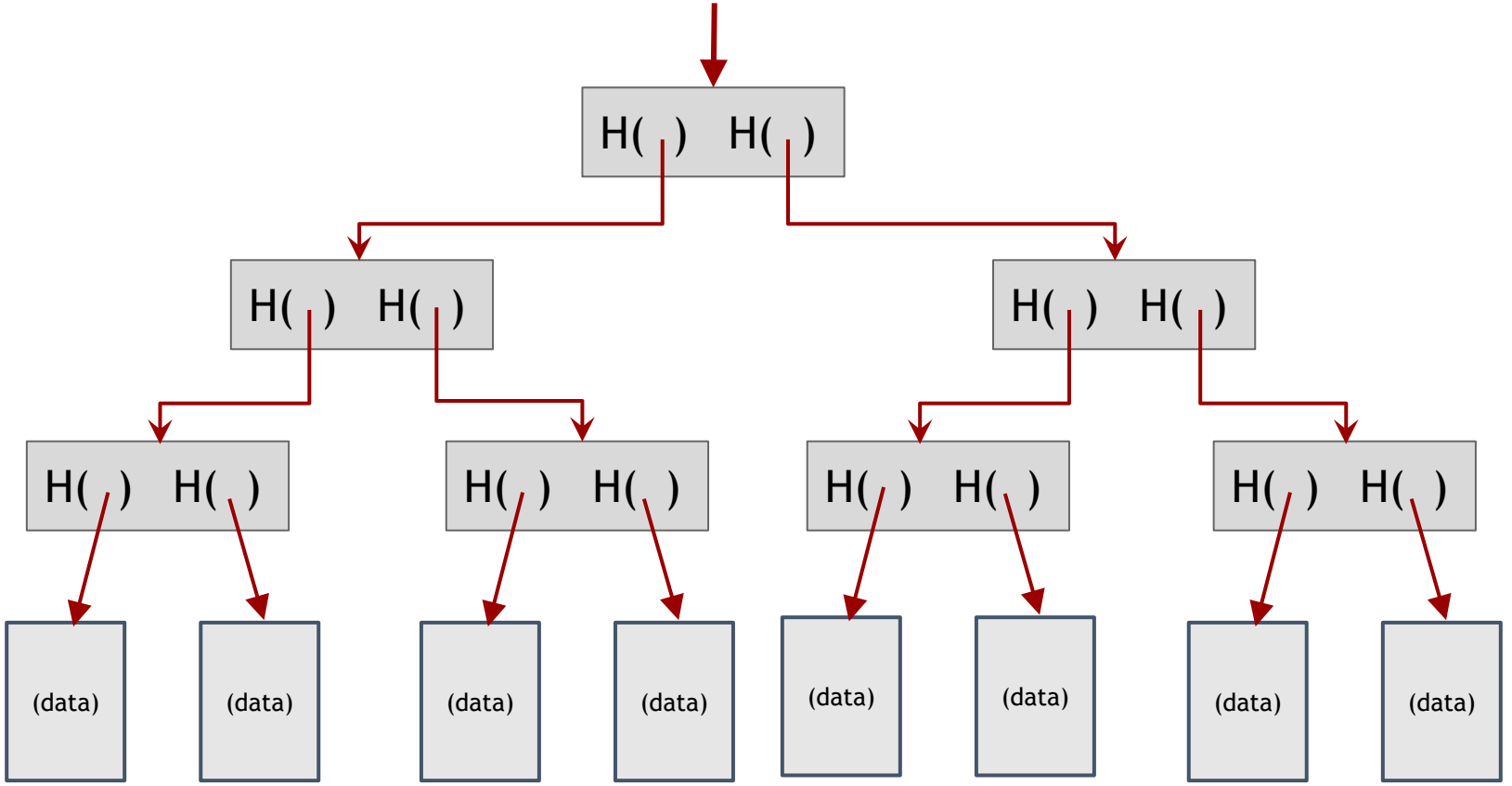
Otherwise, found a hash collision!

Blockchain: Append-only hash chain



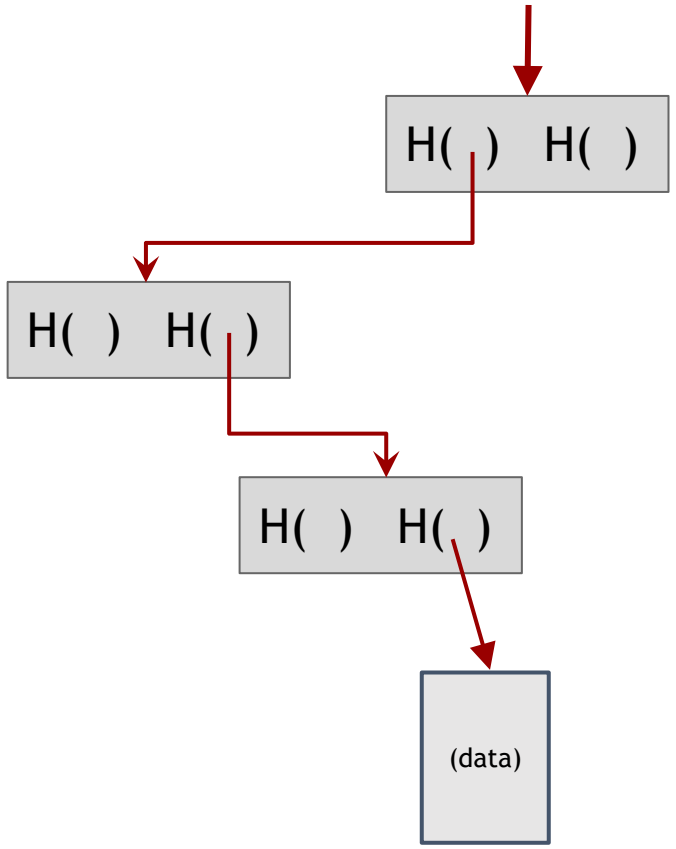
- Hash chain creates “tamper-evident” log of txns
- Security based on collision-resistance of hash function
  - Given m and h = hash(m), difficult to find m’ such that h = hash(m’) and m != m’

Merkle tree



Binary tree with hash pointers

proving membership in a Merkle tree



show O(log n) items

What we want from signatures

Only you can sign, but anyone can verify

Signature is tied to a particular document

can’t be cut-and-pasted to another doc

API for digital signatures

- (sk, pk) := generateKeys(keysize)
  sk: secret signing key
  pk: public verification key
- sig := sign(sk, message)
- isValid := verify(pk, message, sig)

can be randomized algorithms

Requirements for signatures

“valid signatures verify”

verify(pk, message, sign(sk, message)) == true

“can’t forge signatures”

adversary who:

knows pk

gets to see signatures on messages of his choice

can’t produce a verifiable signature on another message

Useful trick: public key == an identity

if you see *sig* such that *verify(pk, msg, sig)==true*,

think of it as

pk says, “*[msg]*”.

to “speak for” *pk*, you must know matching secret key *sk*