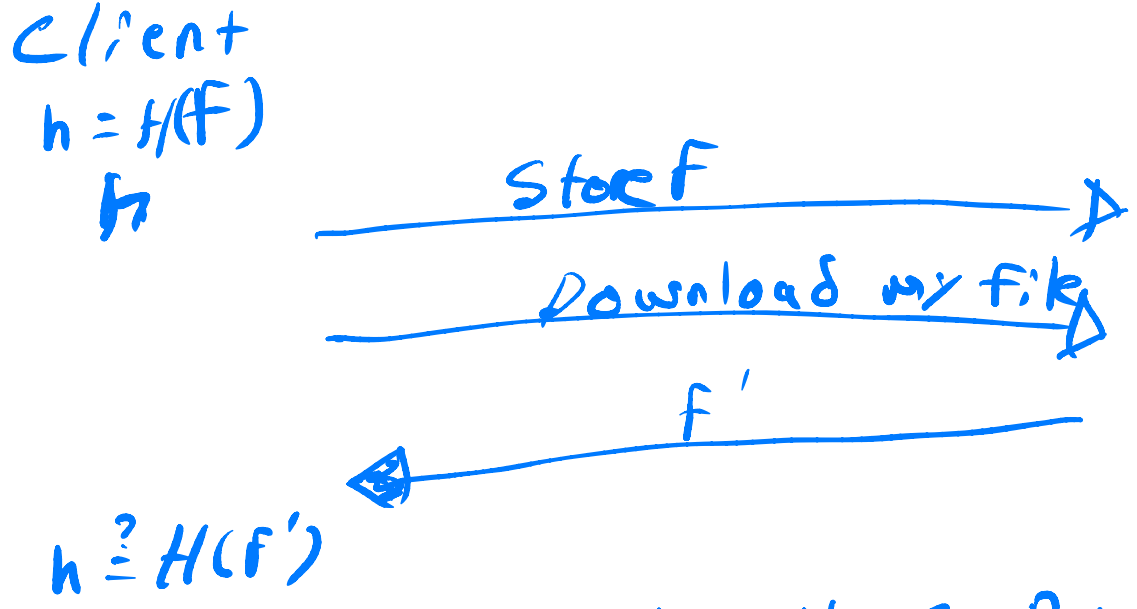
- Collision-resistant hashing + ROM
- Merkle Trees
- Append-only logs
- proof-of-work

Collision-resistant hashing

Amazon



Sufficient: H C-R:

No efficient A can find

$x_1 \neq x_2$ s.t. $H(x_1) = H(x_2)$

except w/ very small prob

Instantiations:

- SHA-256 [US gov't standard]
  - secure hash algorithm
- SHA-3 (Keccak)
- Blake2

All very fast

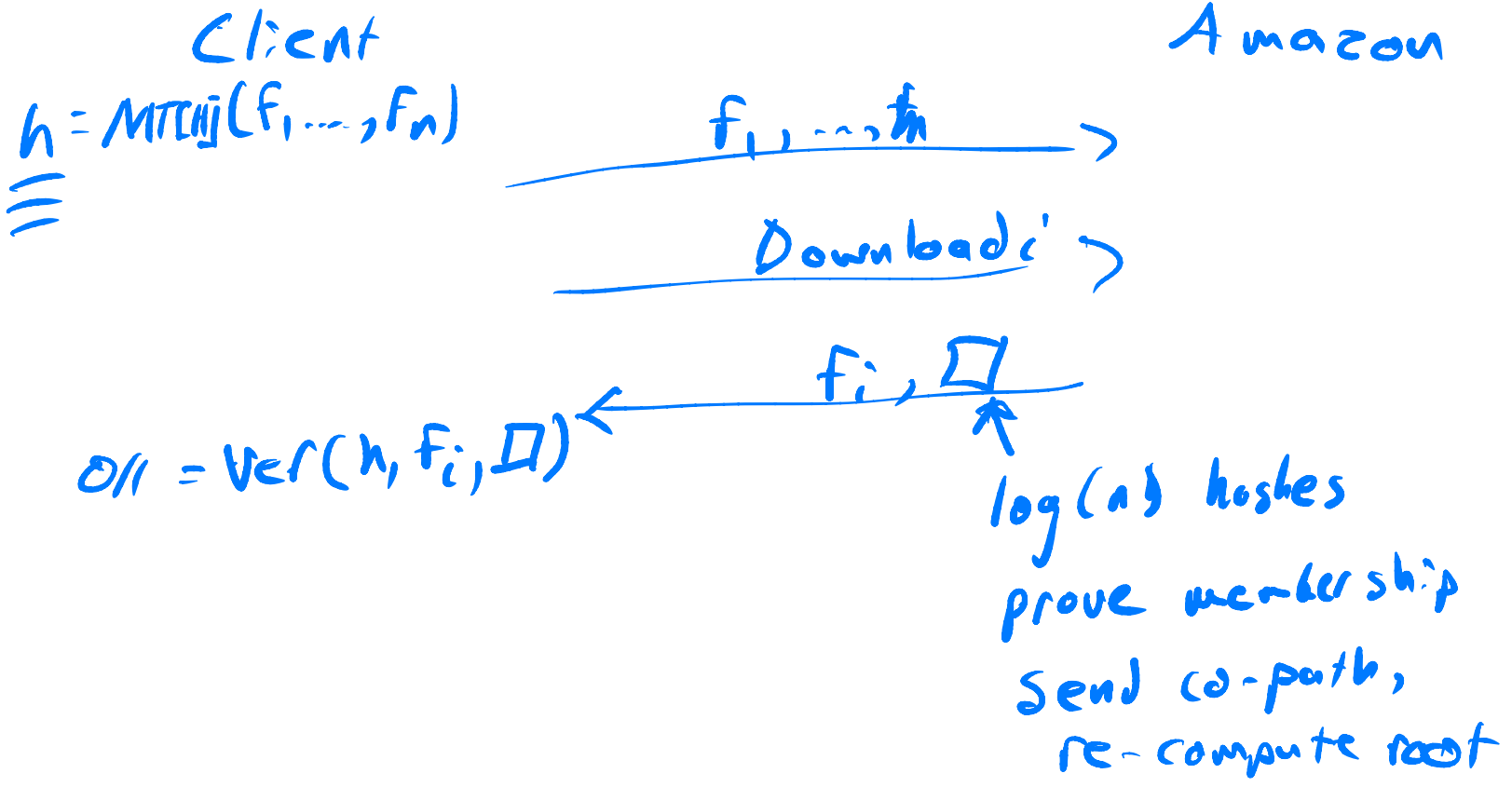
Random oracle model

- Every party has access to shared random function $\{0,1\}^* \to \{0,1\}^n$
- Design in ROM, instantiate w/ concrete hash like SHA-256

Controversial, but usually works fine.

[indifferentiability]

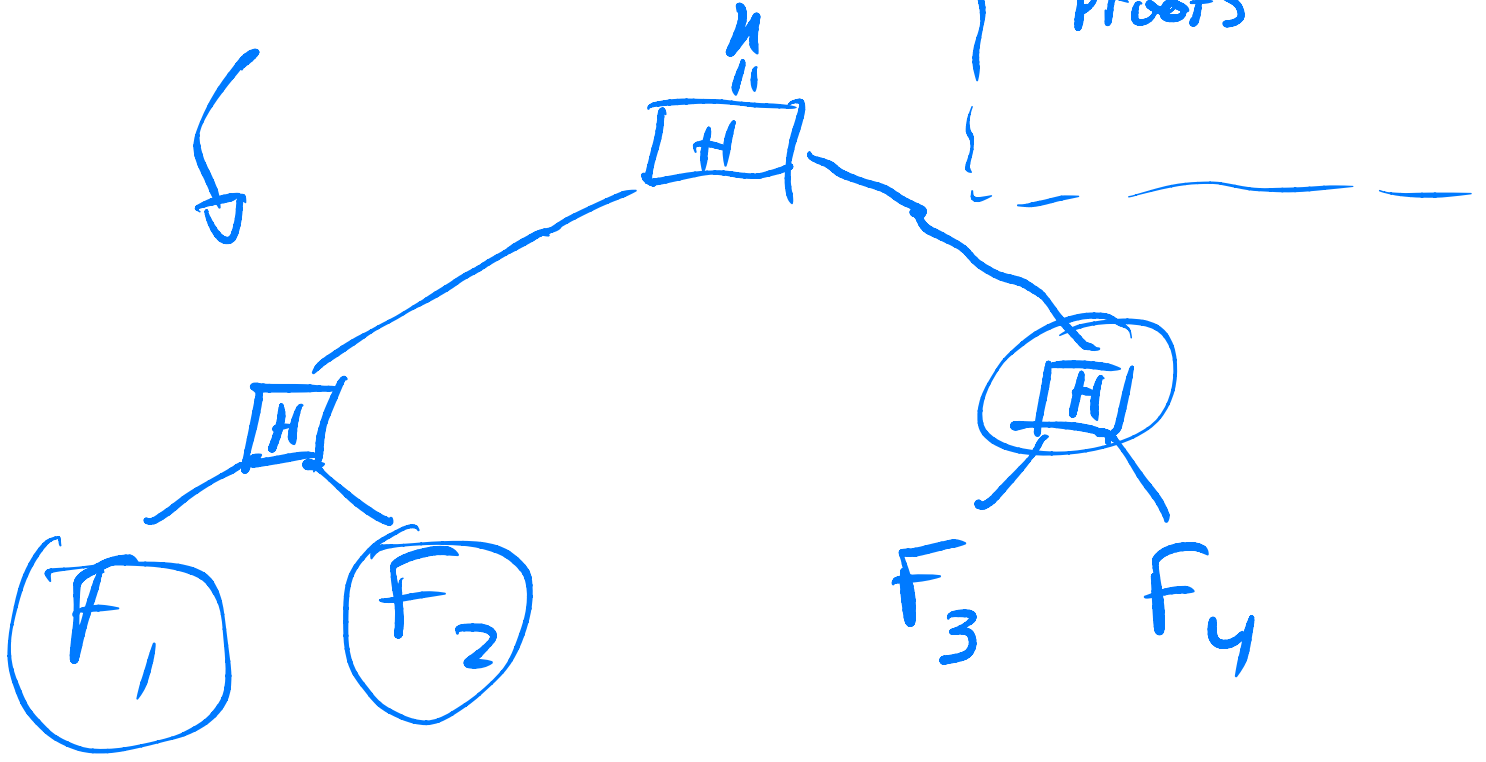
Merkle tree


Client
Amazon
h = MTCHK(f1, ..., fn)
f1, ..., fn
Download i
fi, □
OK = Ver(h, fi, □)
log(n) hashes
prove membership
Send co-path,
re-compute root

$H: 2n \to n$ bits

Merkle tree $MT[H](F_1, \ldots, F_4)$

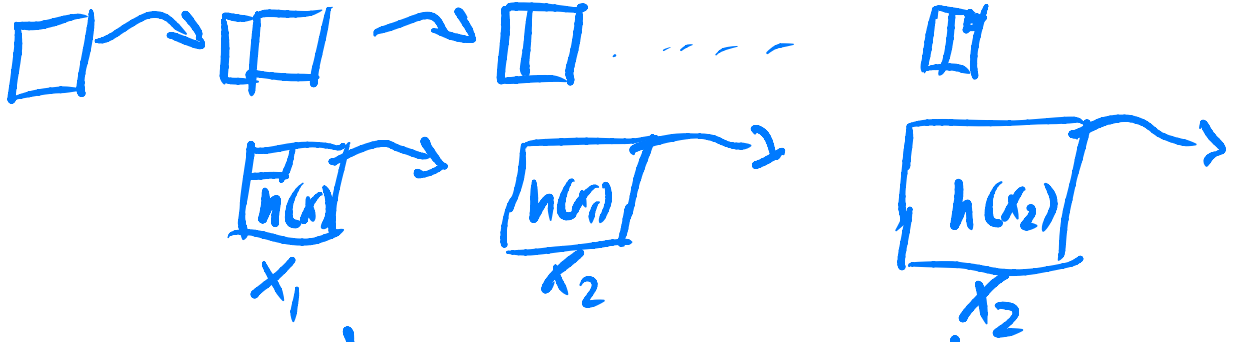
| constant storage
log(n)-size proof
non-inclusion proofs



Authenticated data structures

- Any pointer-based data structure can be authenticated

Append-only log (authenticated linked list)



Given head, anyone can append.

Changing old entries is hard!

Proof-of-work

Cheap-to-verify proof that "lots" of work "was done"

```
def mine(block B, difficulty d):
    n = 0
    while True:
        if H(B, n) < 2^(256-d)
            Ret n
        n += 1
```

If H ROM, proof of "expected" 2^d work. Verify w/ one hash

- progress-free and fair