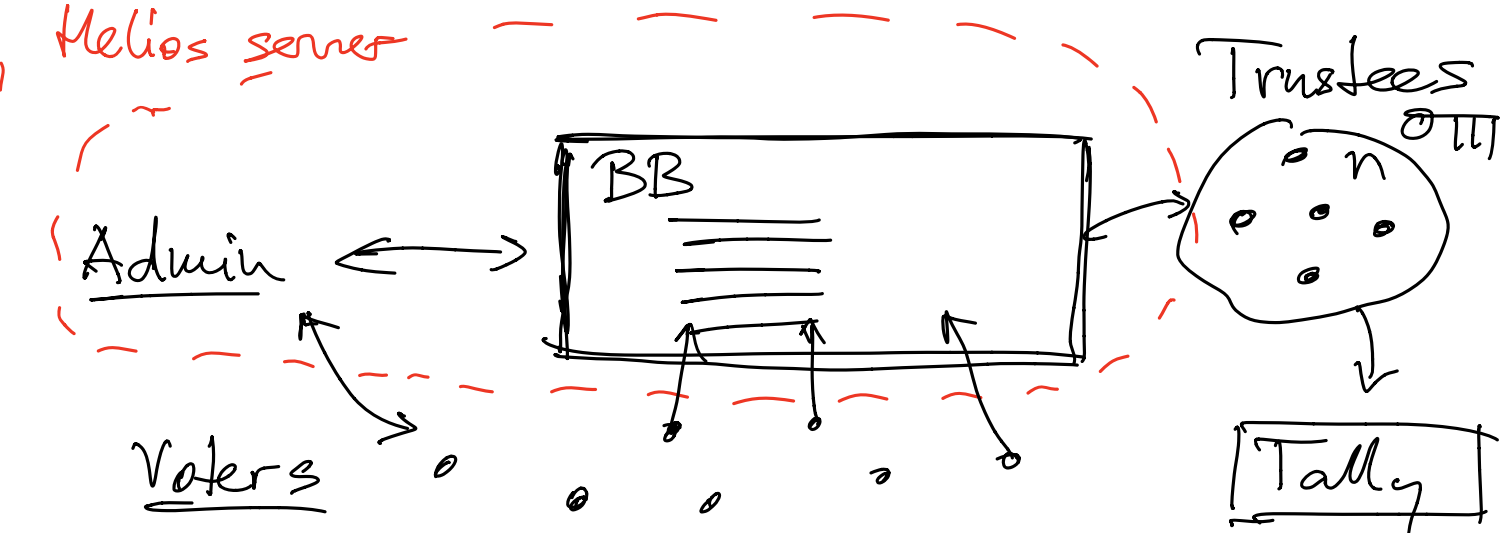
CP , 19.5.21

Helios - A practical online voting system

heliosvoting.org

Model

- n Trustees
- Any number of clients
- Bulletin board for communication
  - implemented by a web server



Approach

- Admin sets up a vote
- Trustees share decryption key using n-of-n sharing
  (could also use t-of-n sharing, but no DKG protocol is implemented)
- Voters encrypt votes
- Voters post encrypted ballots & authenticate to BB
- ZKP for
  - proving that a vote is legitimate
  - proving that decryption is correct

Election audits

1) Cast-as-intended verification

(individual verification) (against malicious local code)

- Voter encrypts ballot (perhaps with mock data)
- To audit, open ballot and verify encryption using an independent platform
- Repeat as often as desired
- Then post to BB (authenticate)

2) Recorded-as-cast verification

(individual) (against corrupted BB)

- BB allows to verify the presence of a ballot with encrypted vote
- Potentially authenticate info on BB

3) Counted-as-recorded verification

(universal)

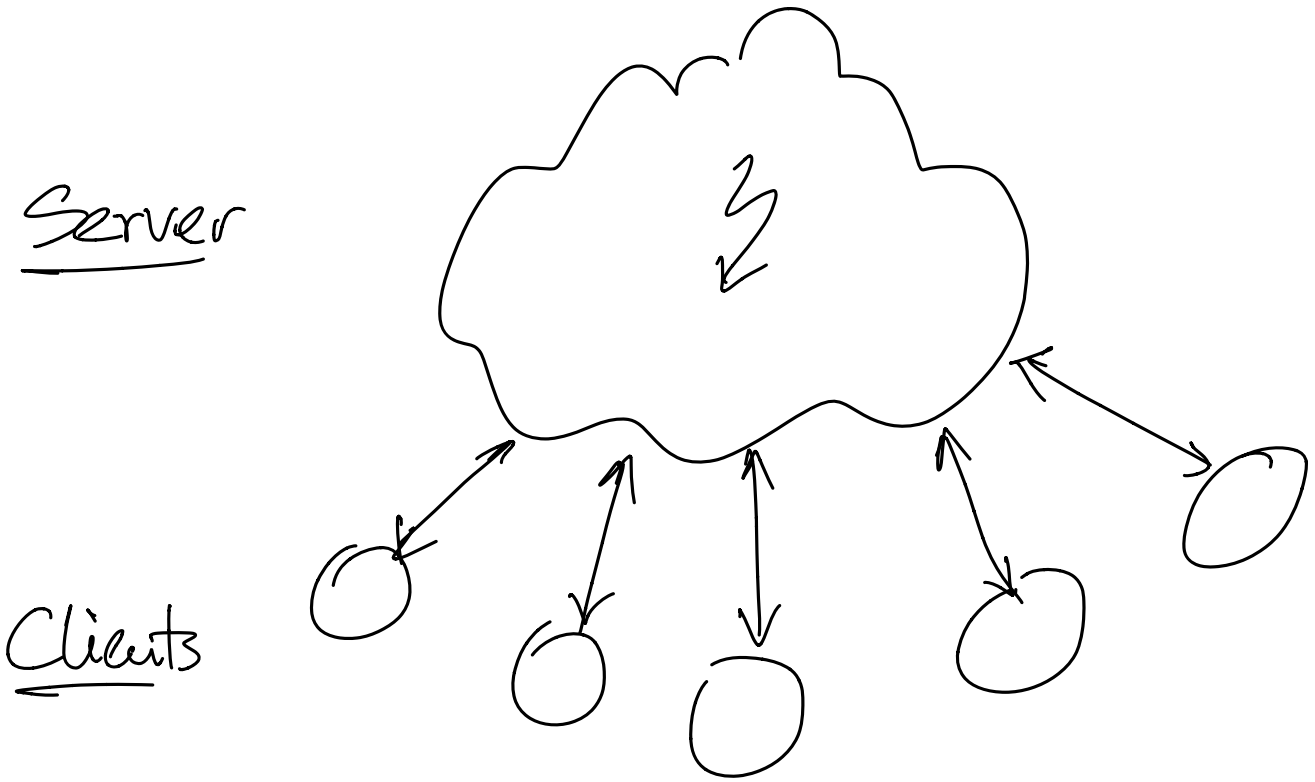
(against corrupted trustees)

- Check that voters are authorized on BB

- Check that result is computed correctly
  - ZKP for correctness of decryption share
  - Proof of correctness for decryption op.

... can be verified by anyone.

10) Integrity Verification



Verify responses from server

Assumptions:

- Clients are correct
- Server is usually correct, but sometimes no

Goal: check responses are always correct

Multiple models

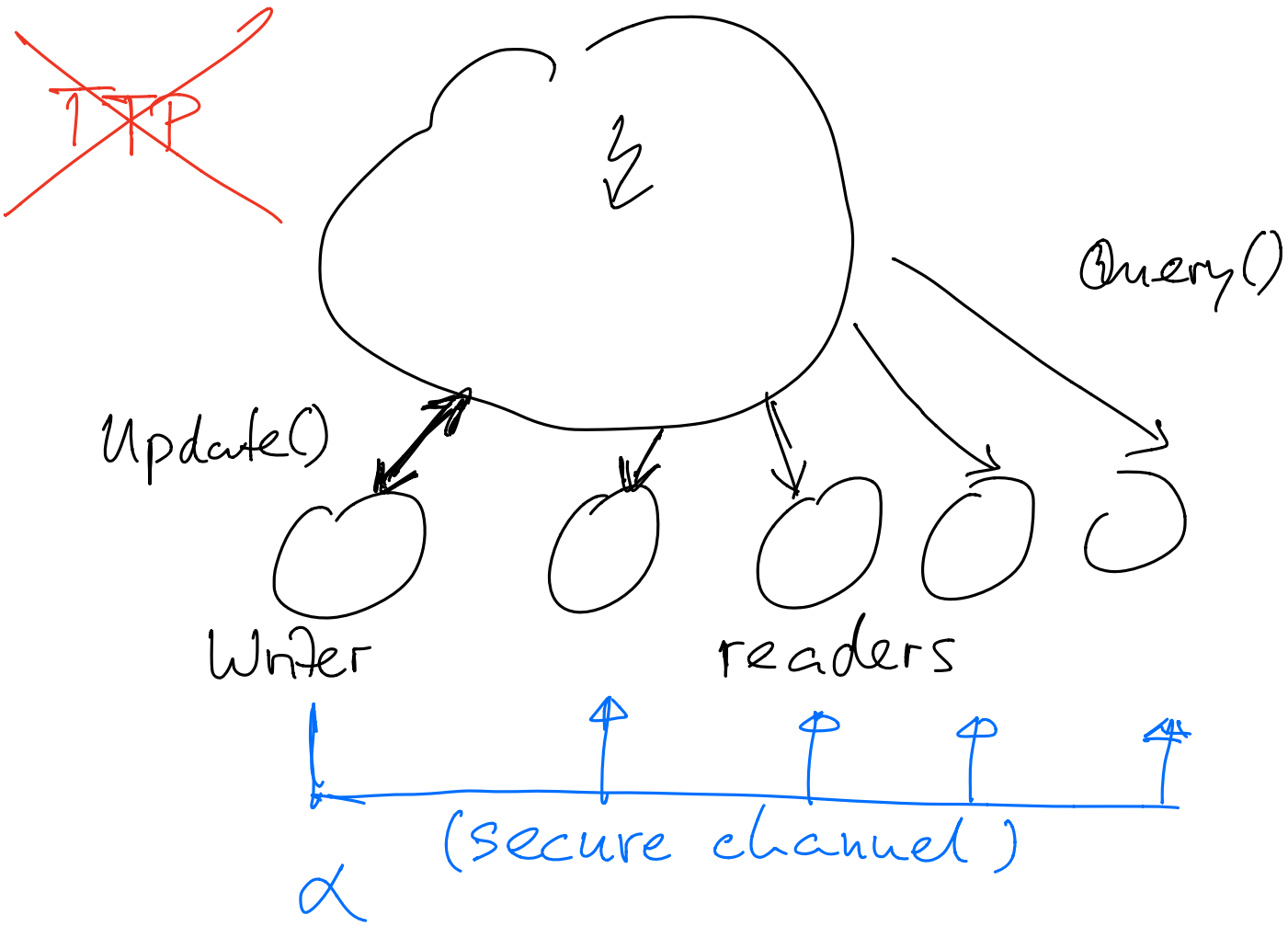
- Server function
  - Storage (write/read)
  - Database (CRUD)
  - Arbitrary (given as circuit)

- Number of clients
  - Single client
  - Multiple clients, one writer many readers
    ... distinguish single-writer from multi-writer scenarios

- Server state ✓

10.1) Authenticate data structures

one writing client
many reading clients



Writer updates state
- incrementally
- state is large

Server (untrusted)
- may perform operations

Many readers

- Retrieve data selectively using queries
- Verify authenticity of responses using authenticator value (α)

Potential solutions

- ~~TTP~~
- hash of an obj.
- public-key digital signatures
  - ↳ issue: sign what?
  - ↳ updates: how prevent replay atks?
- authentication with a MAC
  - ↳ granularity
  - ↳ replay attacks
- hash trees (Merkle trees)

Authenticated data structures

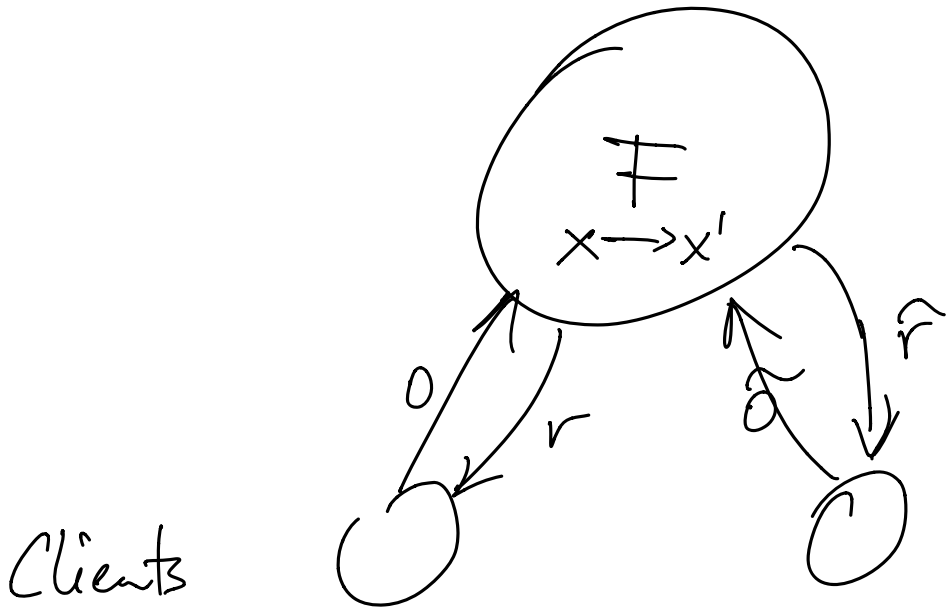
Model

Generic function $F : \mathcal{X} \times \mathcal{O} \to \mathcal{X} \times \mathcal{R}$

Server stores state $x \in \mathcal{X}$

Server executes ops. $o \in \mathcal{O}$

Every op. generates a response $r \in \mathcal{R}$



$F : (x, o) \mapsto (x', r)$

Two kinds of operations:

Updates $U \subseteq \mathcal{O} : \forall u \in U : F(x, u) = (x', \perp)$

Queries $Q \subseteq \mathcal{O} : \forall q \in Q : F(x, q) = (x, r)$

$KeyGen() \rightarrow (sk, pk)$ // Writer

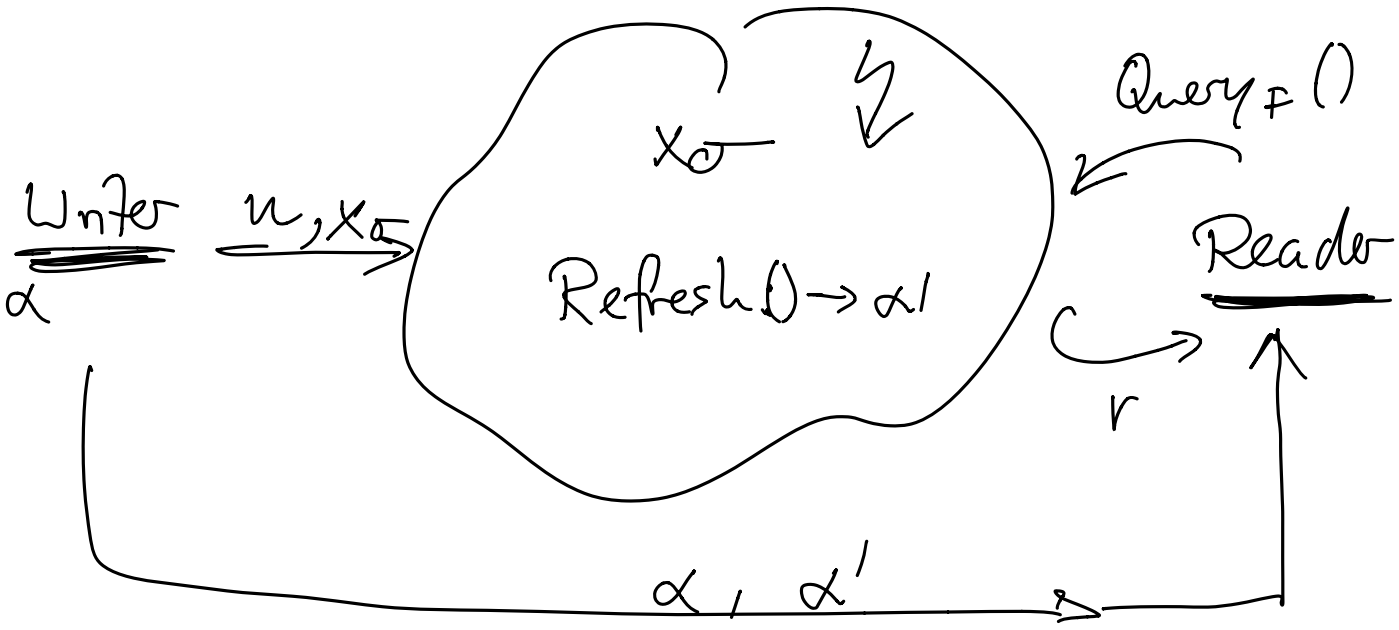
$Init_F(pk, sk, x) \rightarrow (x_\sigma, \alpha)$

// Writer transforms init. state x into protected x_σ and auth. α

$Update_F(pk, sk, x_\sigma, \alpha, u) \rightarrow (\alpha', x'_\sigma)$

update operation $u \in \mathcal{U}$

$Refresh_F(pk, x_\sigma, \alpha, u) \rightarrow (\alpha', x'_\sigma)$



Client ops

Query $(pk, x_0, \alpha, q) \longrightarrow (r, \varphi)$

Verify $(pk, \alpha, q, r) \longrightarrow$ FALSE / TRUE