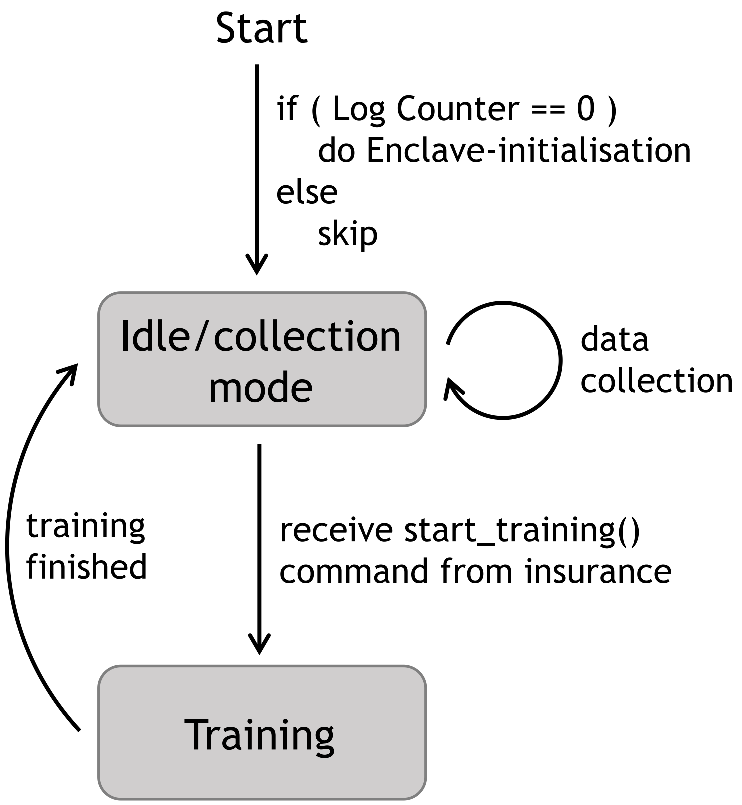


Start

if (Log Counter == 0)
do Enclave-initialisation
else
skip

Idle/collection mode

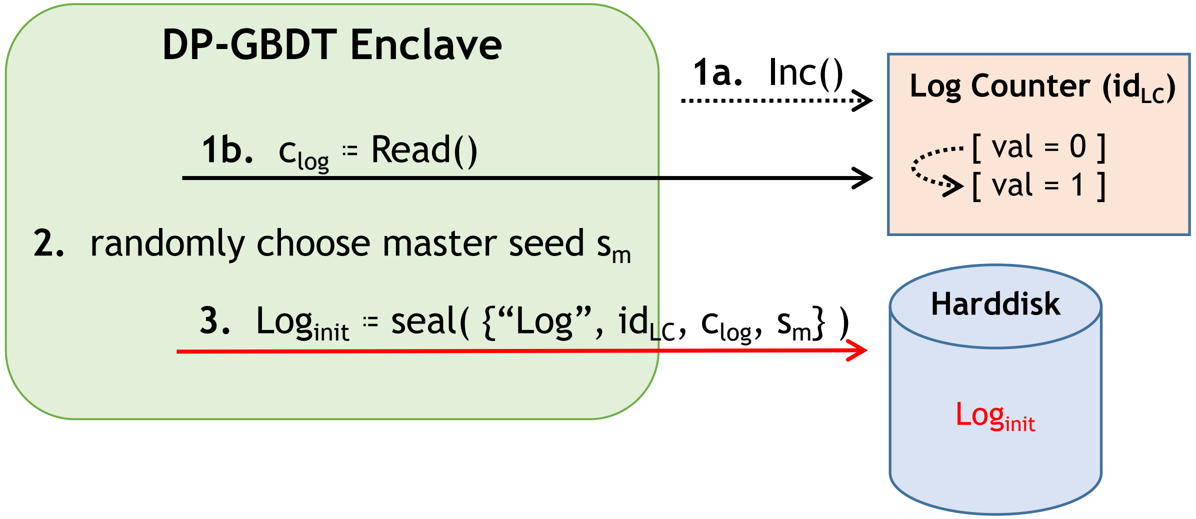
data collection

receive start_training() command from insurance

training finished

Training

Enclave-initialisation:



DP-GBDT Enclave

1a. Inc()

1b. c_{log} ≔ Read()

2. randomly choose master seed s_m

3. Log_{init} ≔ seal({“Log”, id_{LC}, c_{log}, s_m})

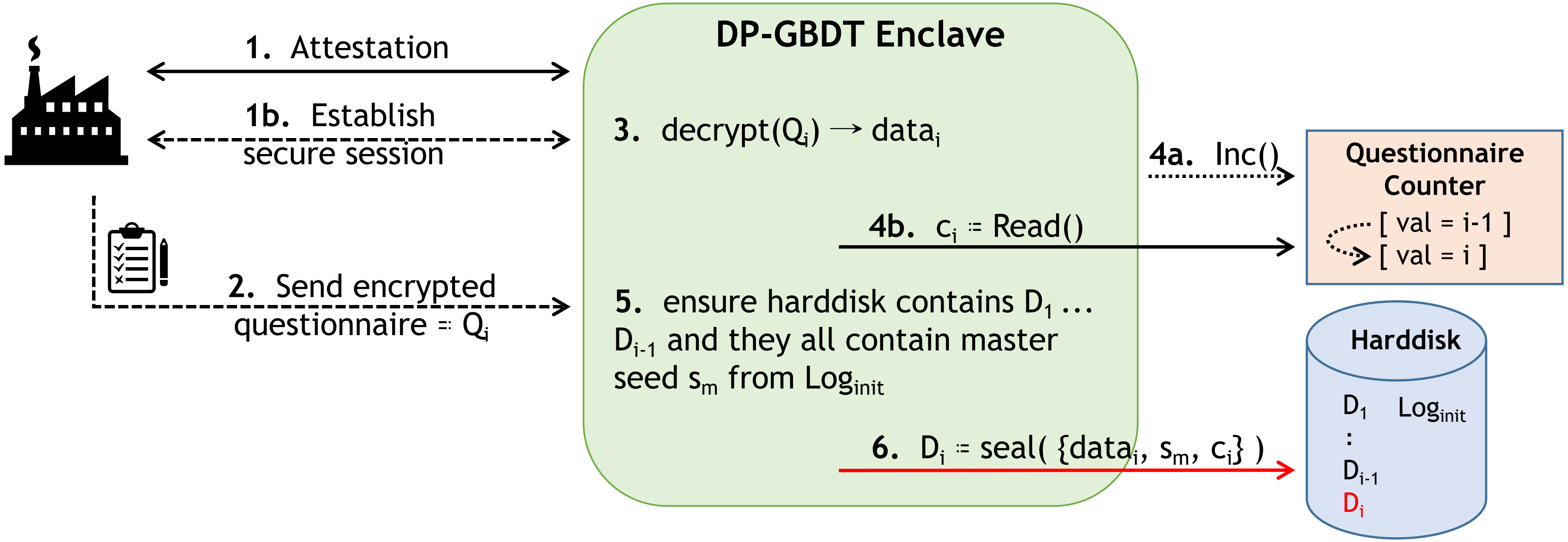
Log Counter (id_{LC})

[val = 0]
[val = 1]

Harddisk

Log_{init}

Data collection



DP-GBDT Enclave

1. Attestation

1b. Establish secure session

2. Send encrypted questionnaire $:= Q_i$

3. decrypt(Q_i) → $data_i$

4a. Inc()

4b. $c_i := \text{Read}()$

5. ensure harddisk contains D_1 … D_{i-1} and they all contain master seed s_m from Log_{init}

6. $D_i := \text{seal}(\{data_i, s_m, c_i\})$

Questionnaire Counter

[val = i-1]

[val = i]

Harddisk

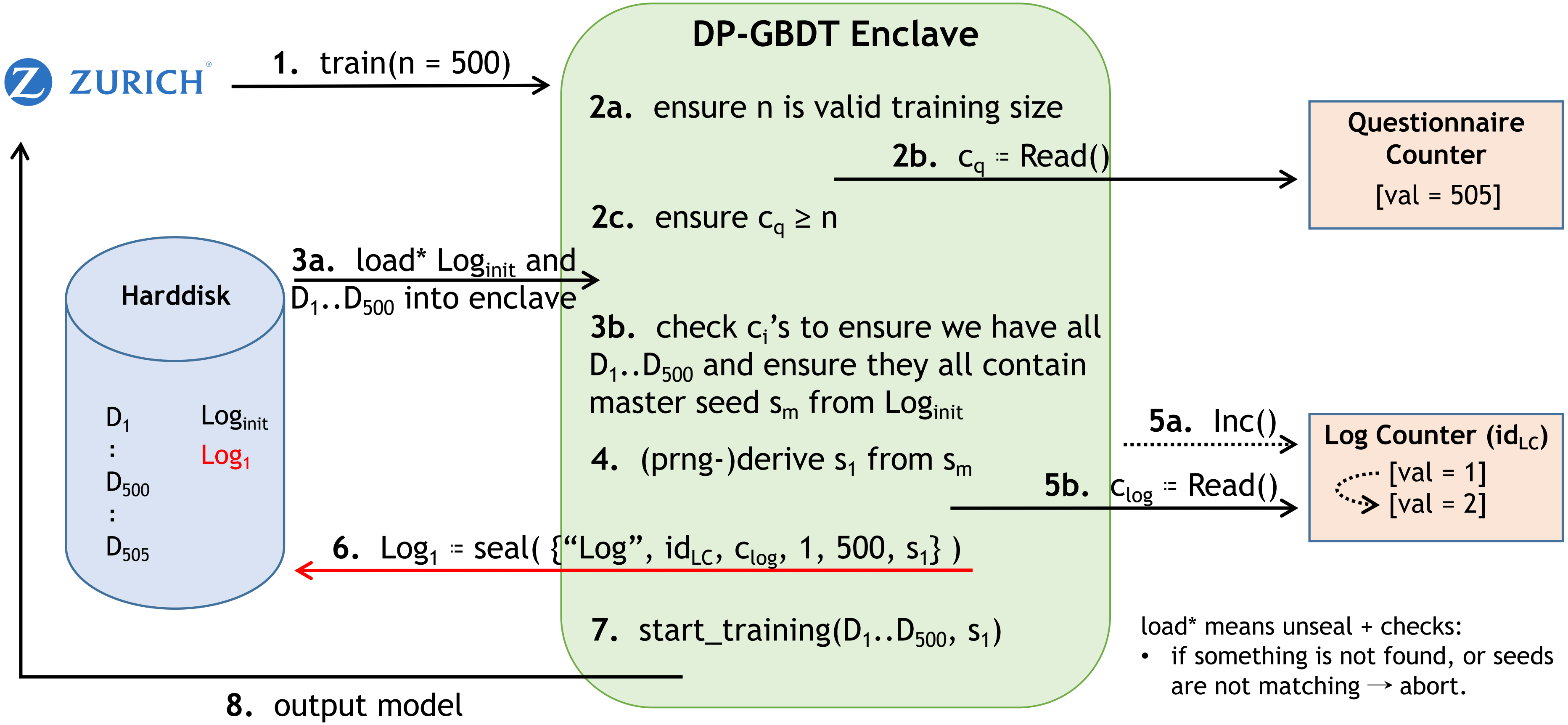
D_1 Log_{init}

:

D_{i-1}

D_i

First training



ZURICH

Harddisk

D_1 : D_{500} : D_{505}

Log_{init}

Log_1

Questionnaire Counter [val = 505]

Log Counter (id_{LC}) [val = 1] [val = 2]

DP-GBDT Enclave

1. train(n = 500)

2a. ensure n is valid training size

2b. c_q ≔ Read()

2c. ensure $c_q \geq n$

3a. load* Log_{init} and $D_1..D_{500}$ into enclave

3b. check c_i’s to ensure we have all $D_1..D_{500}$ and ensure they all contain master seed s_m from Log_{init}

4. (prng-)derive s_1 from s_m

5a. Inc()

5b. c_{log} ≔ Read()

6. Log_1 ≔ seal({“Log”, id_{LC}, c_{log}, 1, 500, s_1})

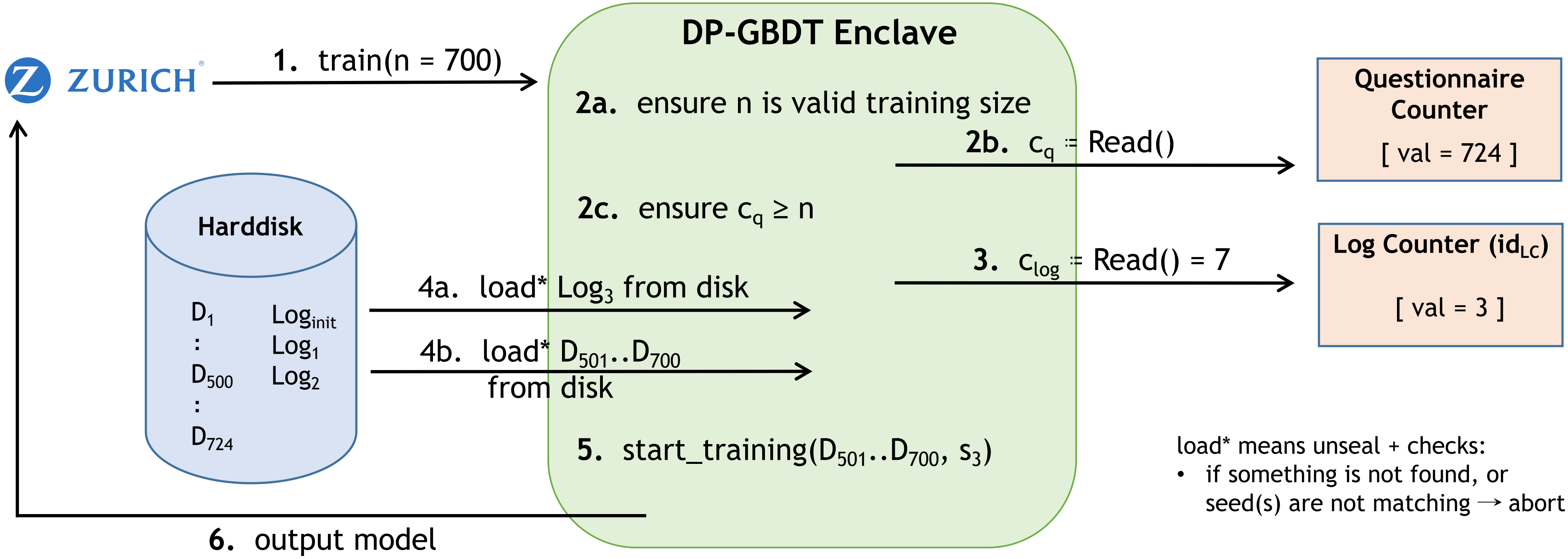
7. start_training($D_1..D_{500}$, s_1)

8. output model

load* means unseal + checks:

- if something is not found, or seeds are not matching → abort.

Re-training (“we lost the model”)



ZURICH

DP-GBDT Enclave

1. train(n = 700)

2a. ensure n is valid training size

2b. c_q ≔ Read()

2c. ensure $c_q \geq n$

3. c_{log} ≔ Read() = 7

4a. load* Log_3 from disk

4b. load* $D_{501}..D_{700}$ from disk

5. start_training($D_{501}..D_{700}$, s_3)

6. output model

Harddisk

D_1
:
D_{500}
:
D_{724}

Log_{init}
Log_1
Log_2

Questionnaire Counter

[val = 724]

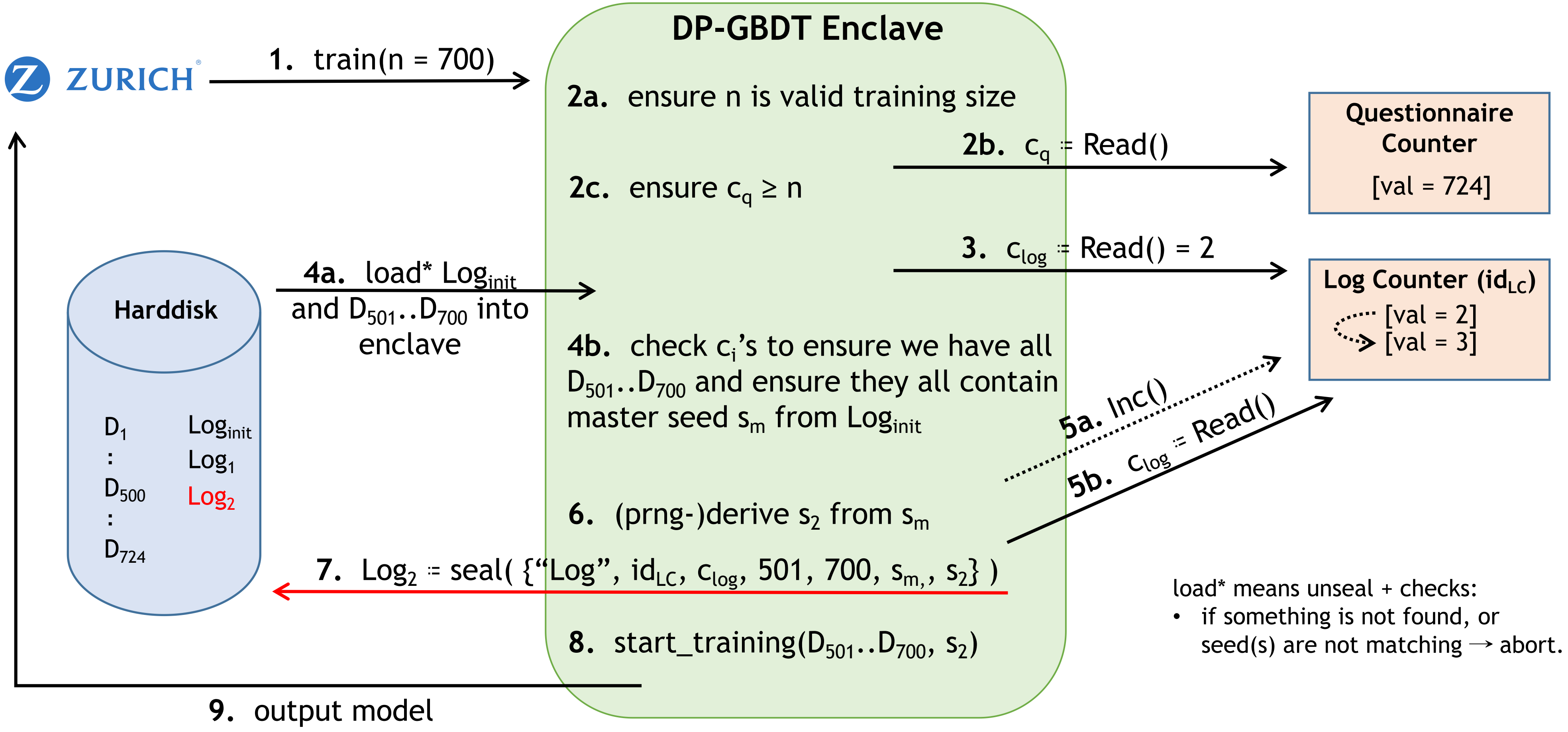
Log Counter (id_{LC})

[val = 3]

load* means unseal + checks:

- if something is not found, or seed(s) are not matching → abort.

Model refinement (add new trees from new samples)



1. train(n = 700)

2a. ensure n is valid training size

2b. $c_q \coloneqq$ Read()

2c. ensure $c_q \geq n$

3. $c_{log} \coloneqq$ Read() = 2

4a. load* Log_{init} and $D_{501}..D_{700}$ into enclave

4b. check c_i's to ensure we have all $D_{501}..D_{700}$ and ensure they all contain master seed s_m from Log_{init}

5a. Inc()

5b. $c_{log} \coloneqq$ Read()

6. (prng-)derive s_2 from s_m

7. $Log_2 \coloneqq$ seal({"Log", id_{LC}, c_{log}, 501, 700, $s_{m,}$, s_2})

8. start_training($D_{501}..D_{700}$, s_2)

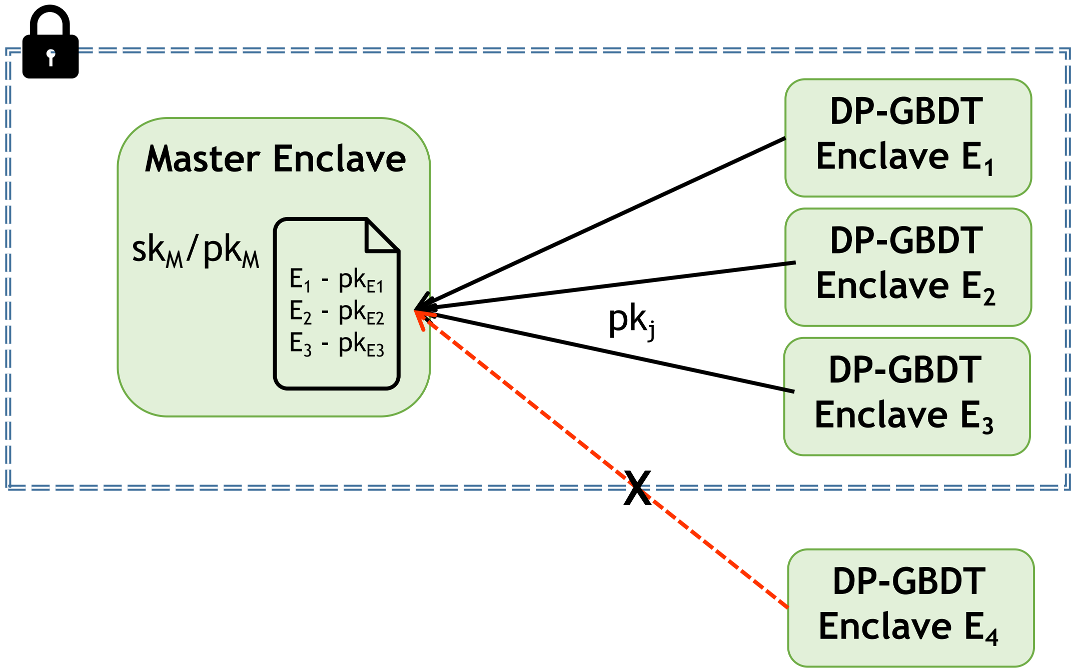
9. output model

load* means unseal + checks:

- if something is not found, or seed(s) are not matching → abort.

Enclave replication

Setup phase:



Master Enclave

sk_M/pk_M

E_1 - pk_{E1}
E_2 - pk_{E2}
E_3 - pk_{E3}

pk_j

DP-GBDT Enclave E_1

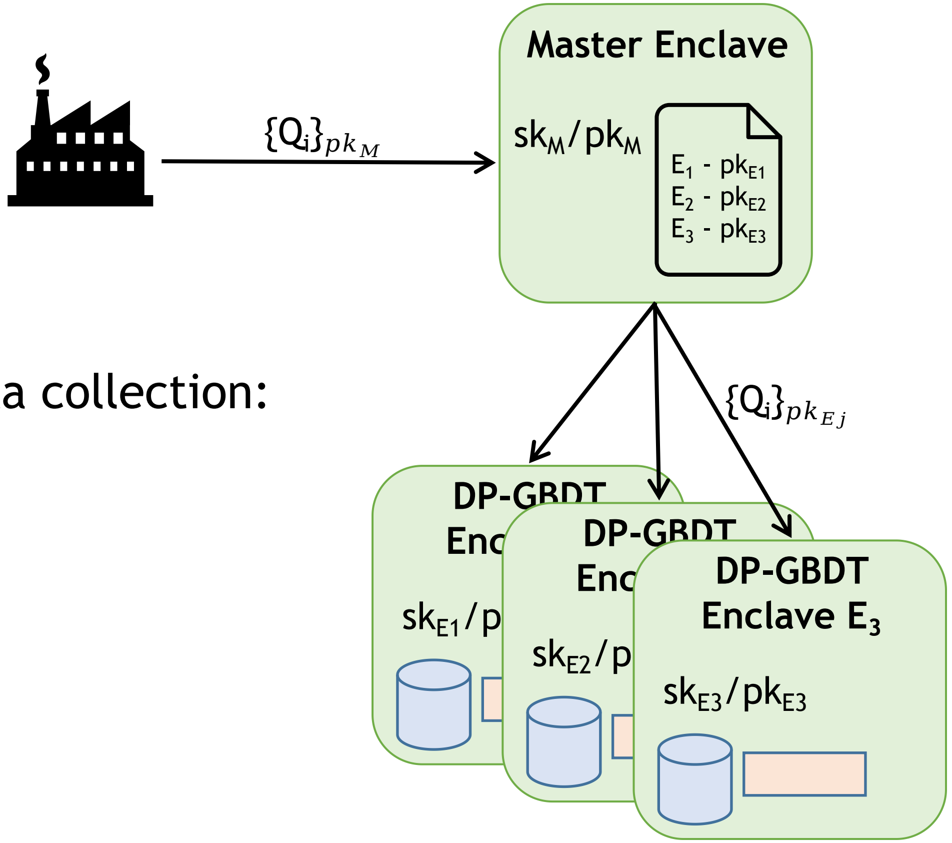
DP-GBDT Enclave E_2

DP-GBDT Enclave E_3

X

DP-GBDT Enclave E_4

Data collection:



$\{Q_i\}_{pk_M}$

Master Enclave

sk_M/pk_M

E_1 - pk_{E1}
E_2 - pk_{E2}
E_3 - pk_{E3}

$\{Q_i\}_{pk_{Ej}}$

DP-GBDT Enc

sk_{E1}/p

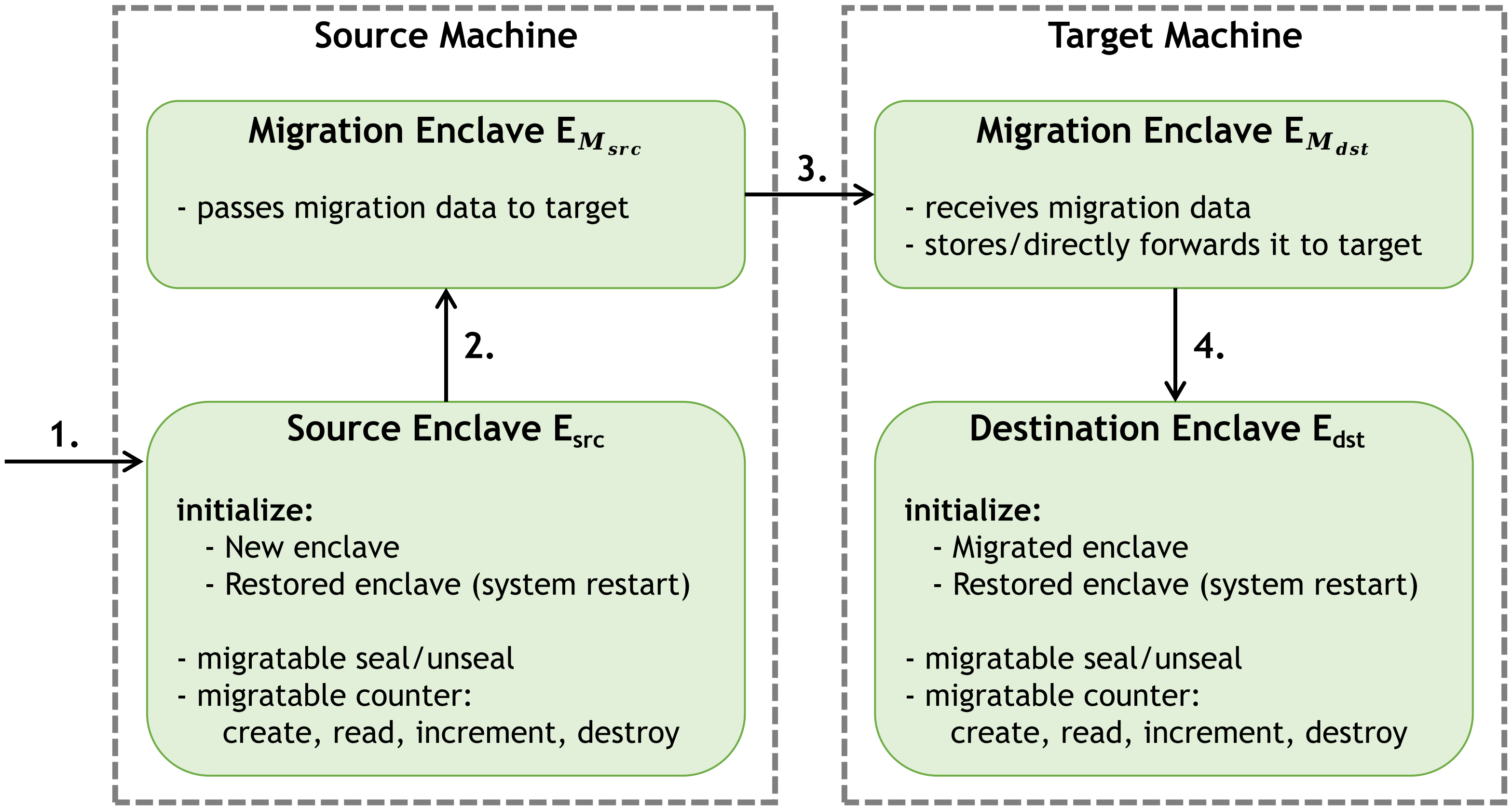
DP-GBDT Enc

sk_{E2}/p

DP-GBDT Enclave E_3

sk_{E3}/pk_{E3}

Enclave migration process



Source Machine

Migration Enclave $E_{M_{src}}$

- passes migration data to target

Source Enclave E_{src}

initialize:

- New enclave
- Restored enclave (system restart)

- migratable seal/unseal
- migratable counter:
  create, read, increment, destroy

Target Machine

Migration Enclave $E_{M_{dst}}$

- receives migration data
- stores/directly forwards it to target

Destination Enclave E_{dst}

initialize:

- Migrated enclave
- Restored enclave (system restart)

- migratable seal/unseal
- migratable counter:
  create, read, increment, destroy