Algorithm 4 Round protocol of a mix server for round r of epoch e.

Input: $state, e, r, roundTimer, isEntry, isExit$

1: **if** $isEntry$ **then**
2: **while** $roundTimer$ not yet fired **do**
3: $recvMsg \leftarrow$ receive from any client that has not yet sent a message this round
4: $k \leftarrow \text{DH}(recvMsg.pk, state.sk_{rcv}^{(e)})$
5: $msg, hasValidTag \leftarrow dec_k(recvMsg.payload)$
6: **if** $hasValidTag$ **then**
7: $state.FirstPool \leftarrow state.FirstPool \cup msg$
8: **end if**
9: **end while**
10: **else**
11: $recvMsgBatch \leftarrow$ allow receive only from preceding mix
12: **for** $i \leftarrow 1 \ldots len(recvMsgBatch)$ **do**
13: $k \leftarrow \text{DH}(recvMsgBatch_i.pk, state.sk_{rcv}^{(e)})$
14: $msg, hasValidTag \leftarrow dec_k(recvMsgBatch_i.payload)$
15: **if** $hasValidTag$ **then**
16: $state.FirstPool \leftarrow state.FirstPool \cup msg$
17: **end if**
18: **end for**
19: **end if**
20: $state.OutPool, state.ThirdPool, state.SecPool \leftarrow state.ThirdPool, state.SecPool, state.FirstPool$
21: $state.FirstPool \leftarrow state.NextPool$ ➤ $state.NextPool$ contains prepared cover messages.
22: $state.SecPool \leftarrow \text{Perm}_{\text{CSPRNG}}(state.SecPool)$ ➤ Break relationships of set indices and receipt times.
23: $state.OutPool \leftarrow state.OutPool \cup \{\, msg_i \in state.SecPool \mid 1 \leq i \leq \lceil (len(state.SecPool)/2) \rceil + r \,\}$
24: $state.OutPool \leftarrow state.OutPool \cup \{\, msg_i \in state.ThirdPool \mid 1 \leq i \leq \lceil (len(state.ThirdPool)/2) \rceil + r \,\}$
25: $state.OutPool \leftarrow \text{Perm}_{\text{CSPRNG}}(state.OutPool)$ ➤ Break relationships of set indices and pool origins.
26: **if** $isExit$ **then**
27: **for** $i \leftarrow 1 \ldots len(state.OutPool)$ **do**
28: $peer, payload \leftarrow state.OutPool_i.pk, state.OutPool_i.payload$
29: send out $payload$ to client $peer$
30: **end for**
31: **else**
32: forward message batch $state.OutPool$ to succeeding mix
33: **end if**
34: $state.NextPool \leftarrow \text{genCoverMsgs}(state.Clients^{(e)})$ ➤ Onion-encrypted (cf. lines 6–18, Alg. 3).