1 Proof of Theorem

Let $\mathcal{H}$ be an adversarial host that plays the **Real** experiment. We now construct an ideal-world adversarial host $\hat{\mathcal{H}}$ such that, for sufficiently large λ and all non-negative q_{forge},

$$\mathbf{Real}(\mathcal{H}, \lambda, q_{\mathsf{forge}}) \overset{c}{\approx} \mathbf{Ideal}(\hat{\mathcal{H}}, \lambda, q_{\mathsf{forge}}).$$

Proof 1 *To prove this statement, we first describe the operation of $\hat{\mathcal{H}}$. We then proceed to demonstrate that if there exists a p.p.t. distinguisher algorithm that distinguishes the output of the two experiments with non-negligible advantage, then one or more of the following are true:* (1) *there exists an adversary that breaks the* SUF-AUTH *security of the ledger authentication tags,* (2) *there is an attack on the binding property of the commitment scheme,* (3) *there exists a distinguisher for the pseudorandom function* PRF*,* (4) *there is an adversary that succeeds against the authenticated encryption scheme, or* (5) *there exists an attacker that breaks the collision-resistance of a hash function.*

The operation of $\hat{\mathcal{H}}$. *The ideal-world adversary $\hat{\mathcal{H}}$ runs the real adversary $\mathcal{H}$ internally, and simulates for it the oracles of the* **Real** *experiment. We will assume that $\hat{\mathcal{H}}$ also has access to an oracle that simulates the real ledger and produces authentication tags, in addition to a* ledger forgery *oracle that allows for the production of "forged" ledger signatures (see §II-B for a discussion of this model.)*

$\hat{\mathcal{H}}$ runs the **Ideal** *experiment. Internally, it maintains the tables T_{enclave}, and T_{forge} as well as a view of the ledger L. The table T_{enclave} contains entries of the following form:*

$$(P, i, \mathsf{l}_i, r_i, \mathcal{S}_{in}, \mathcal{S}_{out}, \mathsf{post}, \mathsf{O}, \mathsf{Pub}, \mathsf{CID})$$

At the start of the experiment, our simulation generates $\mathsf{pp} \leftarrow \mathsf{CSetup}(1^{\lambda})$*. Each query $\mathcal{H}$ makes to the* **Real** *oracles is handled as follows:*

The Ledger Oracle. *When $\mathcal{H}$ queries the ledger oracle on* $(\mathsf{Data}, \mathsf{CID})$*, $\hat{\mathcal{H}}$ executes* Ledger.Post *algorithm (see §II-B). It stores* $(\mathsf{Data}, \mathsf{CID}, \sigma)$ *in a table, and returns* (post, σ) *to $\mathcal{H}$. If two distinct outputs share the same value* post.Hash*, the simulation aborts with an error.*

The Forgery Oracle. *This oracle responds to at most q_{forge} queries made by $\mathcal{H}$. When $\mathcal{H}$ queries on an arbitrary string S, $\hat{\mathcal{H}}$ queries the authentication oracle for a tag σ and records the pair (S, σ) in T_{forge}.*

The Enclave Oracle. *When $\mathcal{H}$ queries the enclave oracle on input* $(P, i, \mathcal{S}_i, \mathsf{l}_i, r_i, \sigma_i, \mathsf{post}_i)$*, $\hat{\mathcal{H}}$ performs the following steps:*

1. *It performs all of the publicly-verifiable checks in the* ExecuteEnclave *algorithm (*i.e.*, all checks that do not rely on the secret K.) It will abort and output a defined error message if $\mathcal{H}$ outputs an authenticator/message pair that was not produced by the* **Ledger** *or* **Forgery** *oracles, a hash collision in* H *or* $\mathsf{H_L}$*, or a collision in the commitment scheme.* $\mathsf{Event_{hashcoll}}$.
2. *It verifies that* (post, σ) *has either been posted to the ledger L, or exists in T_{forge}.*
3. *If $\mathcal{S}_i = \epsilon$ it looks up* CID *in the table T_{enclave} and aborts if any matching entry is found.*
4. *It searches T_{enclave} for an entry where $\mathcal{S}^*_{out} = \mathcal{S}_i$ and aborts if no result is found. Otherwise it obtains the tuple* $(P^*, i^*, \mathsf{l}^*_i, {r_i}^*, \mathcal{S}^*_{in}, \mathcal{S}^*_{out}, \mathsf{post}^*, \mathsf{O}^*, \mathsf{Pub}^*, \mathsf{CID}^*)$ *in T_{enclave} where $\mathcal{S}^*_{out} = \mathcal{S}_i$.*
5. *It checks that $i = i^* + 1$.*
6. *It checks that* $\mathsf{CID} = \mathsf{post.CID} = \mathsf{post}^*.\mathsf{CID}$.
7. *It checks that* $\mathsf{post.PrevHash} = \mathsf{post}^*.\mathsf{Hash}$.
8. *It checks that* $\mathsf{post.Pub} = \mathsf{Pub}^*$.
9. *It checks that $P = P^*$.*

If any of the above checks fail, $\hat{\mathcal{H}}$ aborts and outputs $\perp$ to $\mathcal{H}$.

10. *$\hat{\mathcal{H}}$ first samples $k_{i+1} \leftarrow \{0,1\}^\ell$ uniformly at random, and computes a "dummy" ciphertext*

$$\mathcal{S}'_{out} \leftarrow \mathsf{Encrypt}(k_{i+1}, (1^{\mathsf{Max}(P)}, 1^\ell)).$$

$\hat{\mathcal{H}}$ now selects one of the following three cases. The first case handles repeated *inputs that have already been previously seen. The second case handles inputs that contain valid authenticators from the* **Ledger** *oracle. The final case handles inputs that have been authenticated by the* **Forge** *oracle.*

11. **Process repeated program inputs.** *Whenever $\hat{\mathcal{H}}$ receives an input, it searches its table for any entry that matches $(P, i, \mathsf{I}_i, \mathcal{S}_i, \mathsf{post}_i)$. (Note that we explicitly exclude σ_i from this check.*[1]*) If a matching entry exists, it obtains $(\mathsf{O}, \mathcal{S}_{\mathsf{out}}, \mathsf{Pub}_{\mathsf{out}})$ from the same entry in the table and outputs the tuple $(\mathcal{S}_{out}, \mathsf{O}, \mathsf{Pub}_{\mathsf{out}})$ to $\mathcal{H}$ and halts.*
12. **Process new, and unforged inputs.** *Otherwise, if $(\mathsf{post}_i, \sigma_i)$ is contained within the table maintained by the* **Ledger** *oracle, then $\hat{\mathcal{H}}$ looks up $\mathsf{post}_i.\mathsf{Hash}$ in the* **Ledger***'s table to obtain CID. If this produces multiple possible matches, it aborts and outputs $\mathsf{Event}_{\mathsf{ledgerrepeat}}$.*

    Next, $\hat{\mathcal{H}}$ calls the **Compute** *oracle on $(P, \mathsf{I}_i, \mathsf{CID})$ to obtain $(\mathsf{O}', \mathsf{Pub}')$. $\hat{\mathcal{H}}$ now stores $(P, i, \mathsf{I}_i, r_i, \mathcal{S}_i, \mathcal{S}'_{out}, \mathsf{post}, \mathsf{O}', \mathsf{Pub}')$ in T_{enclave}. It returns $(\mathcal{S}'_{out}, \mathsf{O}', \mathsf{Pub}')$ to $\mathcal{H}$.*
13. **Process new, forged inputs.** *Otherwise, if (post, σ) is contained within T_{forge}, then $\hat{\mathcal{H}}$ identifies the call j at which this value was added to the table, and calls the* **Fork** *oracle on input $(P, \mathsf{I}_, \mathsf{CID}, j)$ to obtain $(\mathsf{O}', \mathsf{Pub}')$. As in the previous step, it stores*

$$(P, i||j, \mathsf{I}_i, r_i, \mathcal{S}_i, \mathcal{S}'_{out}, \mathsf{post}, \mathsf{O}', \mathsf{Pub}', \mathsf{CID})$$

in T_{enclave}. It returns $(\mathcal{S}'_{out}, \mathsf{O}', \mathsf{Pub}')$ to $\mathcal{H}$.

Discussion. *Note that there are three main cases in the simulation above. Whenever the adversary queries the enclave on an state/counter that has previously been queried, $\hat{\mathcal{H}}$ simply returns the same output as it did during the previous query (this occurs at step 11). This is possible because the output of the enclave in the protocol is determined solely by the given inputs (if all public checks pass), and thus repeated inputs cause the enclave to produce the same behavior.*

When the adversary queries on a new input that has been posted to the ledger, $\hat{\mathcal{H}}$ queries the ideal **Compute** *oracle to obtain the appropriate output, and generates a simulated ("dummy ciphertext") encrypted state to return to $\mathcal{H}$. Similarly, when $\mathcal{H}$ queries the enclave on a forged (but otherwise correct) ledger output, $\hat{\mathcal{H}}$ queries the* **Fork** *oracle to obtain the correct output and also returns a dummy ciphertext to $\mathcal{H}$.*

Indistinguishability of $\hat{\mathcal{H}}$'s simulation. *Let D be a p.p.t. distinguisher that succeeds in distinguishing $\hat{\mathcal{H}}$'s output in the* **Ideal** *experiment from $\mathcal{H}$'s output in the* **Real** *experiment with non-negligible advantage. We now show that such an adversary violates one of our assumptions above. The proof proceeds via a series of hybrids, where in each hybrid $\mathcal{H}$ interacts as in the* **Real** *experiment. The first hybrid (***Game 0***) is identically distributed to the* **Real** *experiment, and the final hybrid represents $\hat{\mathcal{H}}$'s simulation above. For notational convenience, let* **Adv** [**Game i**] *be D's absolute advantage in distinguishing the output of* **Game i** *from* **Game 0***,* i.e., *the* **Real** *distribution.*

Game 0. *In this hybrid, $\mathcal{H}$ interacts with the* **Real** *experiment.*

[1]Recall that our simulation has already verified that σ_i passes the value passes the check $\mathsf{Assert}\,(\mathsf{Ledger.Verify}(\mathsf{post}_i, \sigma_i))$. Provided that this check succeeds, it is easy to see that the value σ_i has no further influence on the output of $\mathsf{ExecuteEnclave}$.

Game 1 *(***Abort on [adversary-]forged authenticators.***) This hybrid modifies the previous as follows: in the event that $\mathcal{H}$ queries the* **Enclave** *oracle on any pair* (post, σ) *such that* (1) $\mathsf{Ledger.Verify}(\mathsf{post}, \sigma) = 1$, *and yet* (2) *the pair was not the input (resp. output) of a previous call to either the* **Ledger** *or* **Forgery** *oracles, then abort and output* $\mathsf{Event}_{\mathsf{forge}}$. *We note that if $\mathcal{H}$ causes this event to occur with non-negligible probability, then we can trivially use $\mathcal{H}$ to construct an attack on the* SUF-AUTH *security of the contract authenticator with related probability. Since by assumption the probability of such an event is negligible, we bound* $\mathbf{Adv}\,[\,\mathbf{Game\ 1}\,] \leq \nu_1(\lambda)$.

Game 2 *(***Abort on hash collisions.***) This hybrid modifies the previous as follows: if at any point during the experiment $\mathcal{H}$ causes the functions* $\mathsf{H}, \mathsf{H_L}$ *to be evaluated on inputs* $s_1 \neq s_2$ *such that* $\mathsf{H}(s_1) = \mathsf{H}(s_2)$ *or* $\mathsf{H_L}(s_1) = \mathsf{H_L}(s_2)$, *then abort and output* $\mathsf{Event}_{\mathsf{hashcoll}}$. *Under the assumption that the hash functions* $\mathsf{H}, \mathsf{H_L}$ *are collision-resistant, we have that* $|\mathbf{Adv}\,[\,\mathbf{Game\ 2}\,] - \mathbf{Adv}\,[\,\mathbf{Game\ 1}\,]| \leq \nu_2(\lambda)$.

Game 3 *(***Abort on commitment collisions.***) This hybrid modifies the previous as follows: in the event that $\mathcal{H}$ queries the* **Enclave** *oracle at steps i, j where* $C_i = C_j = \mathsf{Commit}(\mathsf{pp}, (i, \mathsf{I}_i, \mathcal{S}_i, P, \mathsf{post}_i.\mathsf{CID}); r_i) = \mathsf{Commit}(\mathsf{pp}, j \| \mathsf{I}_j \| \mathcal{S}_j \| \mathsf{post}_i.\mathsf{CID}; r_j)$ *and yet* $(i, \mathsf{I}_i, \mathcal{S}_i, P_i, \mathsf{post}_i.\mathsf{CID}) \neq (j, \mathsf{I}_j, \mathcal{S}_j, P_j, \mathsf{post}_j.\mathsf{CID})$, *then abort and output* $\mathsf{Event}_{\mathsf{binding}}$. *We note that if the commitment scheme is binding, this event will occur with at most negligible probability, hence* $|\mathbf{Adv}\,[\,\mathbf{Game\ 3}\,] - \mathbf{Adv}\,[\,\mathbf{Game\ 2}\,]| \leq \nu_3(\lambda)$.

Game 4 *(***Duplicate Enclave calls give identical outputs.***) This hybrid modifies the previous as follows: when $\mathcal{H}$ queries the* **Enclave** *oracle repeatedly on the same values* $(P_i, i, \mathsf{I}_i, \mathcal{S}_i, \mathsf{post}_i)$ *(here we exclude σ_i) and the oracle (as implemented in the previous hybrid) does not output $\perp$, replace the response to all repeated queries subsequent to the first query with the same result as the first query. Recall that by definition the* **Enclave** *oracle's* ExecuteEnclave *calculation is deterministic and solely based on the inputs provided above. Thus, repeated queries on the same input will always produce the same output. Hence by definition* $|\mathbf{Adv}\,[\,\mathbf{Game\ 4}\,] - \mathbf{Adv}\,[\,\mathbf{Game\ 3}\,]| = 0$.

Game 5 *(***Abort on colliding ledger hashes.***) This hybrid modifies the previous as follows: when $\mathcal{H}$ calls the* **Enclave** *oracle on two distinct inputs* $(P_i, i, \mathsf{I}_i, \mathcal{S}_i, \mathsf{post}_i, \sigma_i)$ *and* $(P_j, j, \mathsf{I}_j, \mathcal{S}_j, \mathsf{post}_j, \sigma_j)$, *and if the two inputs do* not *represent repeated inputs (according to* **Game 4***), then: if both* $(\mathsf{post}_i, \sigma_i)$ *and* $(\mathsf{post}_j, \sigma_j)$ *are valid outputs of the* **Ledger** *oracle and yet* $\mathsf{post}_i.\mathsf{Hash} = \mathsf{post}_j.\mathsf{Hash}$ *then abort and output* $\mathsf{Event}_{\mathsf{ledgercoll}}$. *By Lemma 1 this event will occur with at most negligible probability if* $\mathsf{H_L}$ *is collision-resistant and the ledger is implemented correctly. This bounds* $|\mathbf{Adv}\,[\,\mathbf{Game\ 5}\,] - \mathbf{Adv}\,[\,\mathbf{Game\ 4}\,]| \leq \nu_4(\lambda)$.

Game 6 *(*$\hat{\mathcal{H}}$ **can always uniquely identify** CID.*) This hybrid modifies the previous as follows: if at step i the adversary $\mathcal{H}$ calls the* **Enclave** *the oracle (as implemented in the previous hybrid) and* (1) *the oracle does not return $\perp$,* (2) *the inputs to the two calls are not identical (this would be excluded by the earlier hybrids), and* (3) *the pair* $(\mathsf{post}_i, \sigma_i)$ *are in the* **Ledger** *table, and* (4) $\mathsf{post}_i.\mathsf{Hash}$ *matches two distinct entries in the* **Ledger** *table, then abort and output* $\mathsf{Event}_{\mathsf{ledgerrepeat}}$. *By Lemma 2 this event will occur with at most negligible probability. This bounds* $|\mathbf{Adv}\,[\,\mathbf{Game\ 6}\,] - \mathbf{Adv}\,[\,\mathbf{Game\ 5}\,]| \leq \nu_5(\lambda)$.

Game 7 *(***Replace the session keys and pseudorandom coins with random strings.***) This hybrid modifies the previous as follows: if* **Enclave** *does not abort or is not called on repeated inputs, then the pair* $(k_{i+1}, \bar{r}_i)$ *is sampled uniformly at random and recorded in a table for later use. Recall that in the preceding hybrid, this pair is generated as* $(k_{i+1}, \bar{r}_i) \leftarrow \mathsf{PRF}_K(\mathsf{post}_i.\mathsf{Hash})$ *where (by the previous hybrids)* $\mathsf{post}_i.\mathsf{Hash}$ *is guaranteed not to repeat. Similarly, the output of each call* $(k_i, \cdot) \leftarrow \mathsf{PRF}_K(\mathsf{post}_i.\mathsf{PrevHash})$ *is also replaced with a random value when the input* $\mathsf{post}_i.\mathsf{PrevHash}$ *has not been queried to* PRF_K *previously, or the appropriate value (drawn from the table) when this value has been previously queried. If* PRF *is pseudorandom then if* $|\mathbf{Adv}\,[\,\mathbf{Game\ 5}\,] - \mathbf{Adv}\,[\,\mathbf{Game\ 4}\,]|$ *is non-negligible, then we can construct an algorithm that succeeds in distinguishing the* PRF *from a random function. This bounds* $|\mathbf{Adv}\,[\,\mathbf{Game\ 7}\,] - \mathbf{Adv}\,[\,\mathbf{Game\ 6}\,]| \leq \nu_6(\lambda)$.

Game 8 *(***Reject inauthentic ciphertexts.)** *This hybrid modifies the previous as follows: if $\mathcal{H}$ queries the* **Enclave** *oracle on an input $\mathcal{S}_i \neq \varepsilon$ such that (1) the oracle does not reject the input, (2) $\mathsf{Decrypt}(k_i, \mathcal{S}_i)$ does not output $\perp$, and yet (3) the pair $(\mathcal{S}_i, k_i)$ was not generated during a previous query to* **Enclave***, then abort and output $\mathsf{Event}_{\mathsf{auth}}$. We note that that in the previous hybrid each key k_i is generated at random, and only used to encrypt one ciphertext. Thus if $\mathcal{H}$ is able to produce a second ciphertext that satisfies decryption under this key, then we can construct an attacker that forges ciphertexts in the authenticated encryption scheme with non-negligible advantage. Because we assumed the AE scheme was secure, we have that $|\mathbf{Adv}[\,\mathbf{Game\ 8}\,] - \mathbf{Adv}[\,\mathbf{Game\ 7}\,]| \leq \nu_7(\lambda)$.*

Game 9 *(***Abort if inputs are inconsistent.)** *This hybrid modifies the previous as follows: on $\mathcal{H}$'s the i^{th} query to the* **Enclave** *oracle, when the input $\mathcal{S}_i \neq \varepsilon$, let $(\mathsf{post.CID}', P', i', \mathsf{Pub}_{i-1}')$ be the inputs/outputs associated with the previous* **Enclave** *call that produced $\mathcal{S}_i$. If (1) the experiment has not already aborted due to a condition described in previous hybrids and (2) if the* **Enclave** *oracle as implemented in the previous hybrid does not reject the input, and (3) any of the provided inputs $(\mathsf{post.CID}, P, i-1, \mathsf{Pub}_{i-1}) \neq (\mathsf{post.CID}', P', i', \mathsf{Pub}_{i-1}')$ differ from those associated with the previous call to the* **Enclave***, abort and output $\mathsf{Event}_{\mathsf{mismatch}}$. By Lemma 3 we have that $|\mathbf{Adv}[\,\mathbf{Game\ 9}\,] - \mathbf{Adv}[\,\mathbf{Game\ 8}\,]| = 0$.*

Game 10 *(***Replace ciphertexts with dummy ciphertexts.)** *This hybrid modifies the previous as follows: we modify the generation of each ciphertext $\mathcal{S}'_{out}$ to encrypt the unary string $(1^{\mathsf{Max}(P)}, 1^{\ell})$. We first note that the length of the (padded) plaintext is identical between this and the previous hybrid, by the definition of the padding function. Thus if $\mathcal{H}$'s output is significantly different between this and the previous hybrid, we can construct an attacker that succeeds against the confidentiality of the authenticated encryption scheme. By the assumption that the AE scheme is secure, we have that $|\mathbf{Adv}[\,\mathbf{Game\ 10}\,] - \mathbf{Adv}[\,\mathbf{Game\ 9}\,]| \leq \nu_8(\lambda)$.*

We note that **Game 10** *is distributed identically to the simulation provided to $\mathcal{H}$ by $\hat{\mathcal{H}}$. By summation over the hybrids we have that $\mathbf{Adv}[\,\mathbf{Game\ 9}\,] \leq \nu_1(\lambda) + \cdots + \nu_8(\lambda)$ and thus is also negligible in λ. This implies that no p.p.t. distinguisher can distinguish the output of* **Real** *and* **Ideal** *experiments with non-negligible advantage.*

Lemma 1 (Uniqueness of Ledger Identifiers) If $\mathsf{H_L}$ is collision resistant and the ledger operates as described in §II-B, then for *non-repeated* inputs (those not caught in **Game 4**) with all but negligible probability no two calls $i \neq j$ to the **Enclave** oracle in **Game 5** will have $\mathsf{post}_i.\mathsf{Hash} = \mathsf{post}_j.\mathsf{Hash}$.

Proof 2 *Let us imagine that with some non-negligible probability ϵ, $\mathcal{H}$ is able to play* **Game 5** *such that two distinct calls $i \neq j$ to the* **Enclave** *oracle have $\mathsf{post}_i.\mathsf{Hash} = \mathsf{post}_j.\mathsf{Hash}$ and yet the two calls were not identified as repeated inputs (according to the conditions of* **Game 4***). We show that this implies a collision in the hash function $\mathsf{H_L}$.*

Let $(P_j, j, \mathsf{I}_j, \mathcal{S}_j, \mathsf{post}_j), (P_i, i, \mathsf{I}_i, \mathcal{S}_i, \mathsf{post}_i)$ be the inputs to the **Enclave** *oracle (we exclude σ). Let*

$$(\mathsf{Data}_i, \mathsf{CID}_i, \mathsf{post}_i.\mathsf{Hash}), (\mathsf{Data}_j, \mathsf{CID}_j, \mathsf{post}_j.\mathsf{Hash})$$

represent $\mathcal{H}$'s input (resp. output) from the distinct calls to the **Ledger** *oracle. We now show that if two such calls produce identical $\mathsf{post}_i.\mathsf{Hash}$, then the inputs to $\mathsf{H_L}$ are not equal with probability related to ϵ.*

If the two calls to **Enclave** *are* not *repeated inputs, then for $i \neq j$ it holds that:*

$$(P_j, j, \mathsf{I}_j, \mathcal{S}_j, \mathsf{post}_j) \neq (P_j, j, \mathsf{I}_j, \mathcal{S}_j, \mathsf{post}_j)$$

Recall as well that:

$$\mathsf{post}_i.\mathsf{Hash} = \mathsf{H_L}(\mathsf{post}_i.\mathsf{Data} \| \mathsf{post}_i.\mathsf{PrevHash})$$

$$\mathsf{post}_j.\mathsf{Hash} = \mathsf{H_L}(\mathsf{post}_j.\mathsf{Data} \| \mathsf{post}_j.\mathsf{PrevHash})$$

And:

$$\mathsf{post}_i.\mathsf{Data} = \mathsf{Commit}(\mathsf{pp}, (i, \mathsf{I}_i, \mathcal{S}_i, P_i); r_i)$$

$$\mathsf{post}_j.\mathsf{Data} = \mathsf{Commit}(\mathsf{pp}, (j, \mathsf{I}_j, \mathcal{S}_j, P_j); r_j)$$

Thus note that if $\mathsf{post}_i.\mathsf{Hash} = \mathsf{post}_j.\mathsf{Hash}$ *then this would imply that the simulation would abort with either* $\mathsf{Event}_{\mathsf{hashcoll}}$ *or* $\mathsf{Event}_{\mathsf{binding}}$*. Since this has not occurred, then the probability of such a collision is* 0*.*

Lemma 2 (No duplicate Ledger identifiers) If $\mathsf{H_L}$ is collision-resistant, then for all $\mathcal{H}$, $\mathbf{Pr}\left[\,\mathsf{Event}_{\mathsf{ledgerrepeat}}\,\right] \leq \nu_6(\lambda)$.

Proof 3 *Let us assume by contradiction that* $\mathcal{H}$ *is able to query the* **Ledger** *oracle on two distinct inputs such that the oracle returns (and records in its table) two distinct transactions* post_i *and* post_j *such that* $\mathsf{post}_i.\mathsf{Hash} = \mathsf{post}_j.\mathsf{Hash}$*. Then we construct a second adversary* $\mathcal{A}$ *that outputs a collision in the hash function* $\mathsf{H_L}$*.* $\mathcal{A}$ *conducts the experiment of* **Game 6** *with* $\mathcal{H}$*.*

Let i, j *be any two integers. If at any point* $\mathcal{H}$*, at the* j^{th} *call to the* **Ledger** *oracle, submits a pair* $(\mathsf{Data}, \mathsf{CID})$ *such that* $\mathsf{post}_j.\mathsf{Hash} = \mathsf{post}_i.\mathsf{Hash}$*, then* $\mathcal{A}$ *terminates and outputs the collision pair* $(\mathsf{post}_i.\mathsf{Data}\|\mathsf{post}_i.\mathsf{PrevHash}), (\mathsf{post}_j.\mathsf{Data}\|\mathsf{post}_j.\mathsf{PrevHash})$*.*

Clearly if $\mathsf{post}_j.\mathsf{Hash} = \mathsf{post}_i.\mathsf{Hash}$ *but*

$$(\mathsf{post}_i.\mathsf{Data}\|\mathsf{post}_i.\mathsf{PrevHash}) \neq (\mathsf{post}_j.\mathsf{Data}\|\mathsf{post}_j.\mathsf{PrevHash})$$

then $\mathcal{A}$ *has identified a collision in* $\mathsf{H_L}$*. It remains only to show that when* $\mathcal{H}$ *succeeds in triggering this event, the latter inequality must hold. Our proof proceeds inductively.*

1. *If* $j = 0$ *then this condition cannot occur, as there is no previous entry in the table.*

2. *If* $j > 0$ *then there are two subconditions:*

   (a) *If* $\mathcal{H}$ *has* not *previously called the* **Ledger** *oracle on* CID *then* $\mathsf{post}_j.\mathsf{PrevHash}$ *is set to a unique identifier based on* CID*. Because there has been no previous call on input* CID*, there cannot exist a second value* $\mathsf{post}_i.\mathsf{PrevHash}$ *in the table that shares the same value. Thus* $\forall i$ *it holds that* $\mathsf{post}_j.\mathsf{PrevHash} \neq \mathsf{post}_i.\mathsf{PrevHash}$ *and thus the main inequality holds.*

   (b) *If* $\mathcal{H}$ *has previously called the* **Ledger** *oracle on input* CID*, then (by the definition of the Ledger interface) there must exist some* i *such that* $\mathsf{post}_j.\mathsf{PrevHash} = \mathsf{post}_i.\mathsf{Hash} = \mathsf{H_L}(\mathsf{post}_i.\mathsf{Data}\|\mathsf{post}_i.\mathsf{PrevHash})$*. We now consider two subcases:*

      i. *If the* i^{th} *call to the* **Ledger** *oracle was the first call made on input* CID*, then by definition* $\mathsf{post}_i.\mathsf{PrevHash} \neq \mathsf{post}_j.\mathsf{PrevHash}$ *because as the root of a new chain,* $\mathsf{post}_i.\mathsf{PrevHash}$ *has a special structure and cannot be equal to the output of* $\mathsf{H_L}$*.*

      ii. *If the* i^{th} *call to* **Ledger** *was* not *the first call on input* CID*, and if* $\mathsf{post}_j.\mathsf{PrevHash} = \mathsf{post}_i.\mathsf{PrevHash}$*, then by definition there must exist a third integer* $k < j$ *such that the* k^{th} *call to the* **Ledger** *oracle was also on input* CID *and* $\mathsf{post}_i.\mathsf{PrevHash} = \mathsf{post}_k.\mathsf{Hash} = \mathsf{H_L}(\mathsf{post}_k.\mathsf{Data}\|\mathsf{post}_k.\mathsf{PrevHash})$*. However, this event cannot occur, as this would imply that* $\mathsf{post}_i.\mathsf{Hash} = \mathsf{post}_k.\mathsf{Hash}$ *and thus* $\mathcal{A}$ *would have already terminated and output a collision prior to reaching this point.*

   In all of the above cases $(\mathsf{post}_i.\mathsf{Data}\|\mathsf{post}_i.\mathsf{PrevHash}) \neq (\mathsf{post}_j.\mathsf{Data}\|\mathsf{post}_j.\mathsf{PrevHash})$ *and so if* $\mathcal{H}$ *triggers this condition, then* $\mathcal{A}$ *finds a collision in* $\mathsf{H_L}$*. Because we assume that* $\mathsf{H_L}$ *is collision-resistant, we can bound the probability of* $\mathsf{Event}_{\mathsf{ledgerrepeat}}$ *to be negligible in* λ*.*

Lemma 3 (Consistency of encrypted state) It holds that for all p.p.t. $\mathcal{H}$, $\mathbf{Pr}\left[\,\mathsf{Event}_{\mathsf{mismatch}}\,\right] = 0$.

Proof 4 *For* $\mathsf{Event}_{\mathsf{mismatch}}$ *to occur, one of the following conditions must occur. First, it must be the case that* $\mathcal{H}$ *has previously called* **Enclave** *on a set of values* $(\mathsf{post.CID}', P', i', \mathsf{Pub}_{i-1}')$*, which produced a state ciphertext* $\mathcal{S}$ *that embeds*

$$H_P = \mathsf{H}(\mathsf{post.CID}'\|P'\|i'\|\mathsf{Pub}'_{i-1}).$$

And simultaneously it must be the case that, on input the state ciphertext $\mathcal{S}$, the assertion $\mathsf{Assert}(H_P = \mathsf{H}(\mathsf{CID}\|P\|i\|\mathsf{Pub}_{i-1}))$ *would not fail and cause the oracle to return $\perp$.*

However, for this event to occur, it would require one of the following to be true: either the state ciphertext would have to be inauthentic, resulting in a previous abort $\mathsf{Event}_{\mathsf{auth}}$ *(or another abort). Or the hash function would have to include a collision, resulting in* $\mathsf{Event}_{\mathsf{hashcoll}}$. *Since we require that the simulation would abort on these other events before it aborts with* $\mathsf{Event}_{\mathsf{mismatch}}$, *this event can never occur.*

Lemma 4 (Uniqueness of PRF inputs) No adversary playing **Game 5** will (with non-negligible probability) call the **Enclave** oracle at steps i, j that (1) the oracle does not return $\perp$ or the experiment aborts in one of the calls, (2) the input values $(P_1, i, \mathsf{I}_1, \mathcal{S}_1, \mathsf{CID}_1)$ and $(P_2, j, \mathsf{I}_2, \mathcal{S}_2, \mathsf{CID}_2)$ to the two oracle calls are distinct, and (3) during these invocations the evaluation $\mathsf{PRF}_K(\mathsf{post}_i.\mathsf{Hash})$ and $\mathsf{PRF}_K(\mathsf{post}_j.\mathsf{Hash})$ uses $\mathsf{post}_i.\mathsf{Hash} = \mathsf{post}_j.\mathsf{Hash}$.

Proof 5 *Let $\mathcal{H}'$ be an adversary that succeeds in triggering the above condition with non-negligible probability ϵ. We now show this violates one of the assumptions given above.*

Let us define the two (partial) input tuples provided to the **Enclave** *oracle as*

$$\mathsf{Input}_1 = (P_1, i, \mathsf{I}_1, \mathcal{S}_1, \mathsf{CID}_1) \text{ and } \mathsf{Input}_2 = (P_2, j, \mathsf{I}_2, \mathcal{S}_2, \mathsf{CID}_2)$$

where $\mathsf{Input}_1 \neq \mathsf{Input}_2$ *(we also separately define* $\mathsf{post}_1, \mathsf{post}_2$ *as inputs to the oracle.) Recall that in the first instance,* PRF *is evaluated on* $(K, \mathsf{post}_i.\mathsf{Hash})$ *and in the second it is evaluated on* $(K, \mathsf{post}_j.\mathsf{Hash})$. *We now argue that if the oracle does not output $\perp$ (and the experiment itself has not aborted), and if* $\mathsf{Input}_i \neq \mathsf{Input}_j$ *then this must imply that* $\mathsf{post}_i.\mathsf{Hash} \neq \mathsf{post}_j.\mathsf{Hash}$.

Note that each of the fields $\mathsf{post}_1.\mathsf{Data}, \mathsf{post}_2.\mathsf{Data}$ *embeds a commitment computed on* $(\mathsf{I}_1, i, \mathcal{S}_1, P_1, \mathsf{CID}_1)$, $(\mathsf{I}_2, j, \mathcal{S}_2, P_2, \mathsf{CID}_2)$ *respectively. If the* **Enclave** *oracle does not abort and the experiment does not abort, then if* $(\mathsf{I}_1, i, \mathcal{S}_1, P_1, \mathsf{CID}_1) \neq (\mathsf{I}_2, j, \mathcal{S}_2, P_2, \mathsf{CID}_2)$ *then it must hold that* (1) $\mathsf{post}_1.\mathsf{Data} \neq \mathsf{post}_2.\mathsf{Data}$ *(if this were not the case then it would imply a violation of the binding property of the commitment scheme, and hence an abort with* $\mathsf{Event}_{\mathsf{binding}}$*), or* (2) $\mathsf{post}_1.\mathsf{Hash} \neq \mathsf{post}_2.\mathsf{Hash}$ *(because each* Hash *is computed by applying* $\mathsf{H_L}$ *to the* Data *field, and a collision in these values would imply an abort with* $\mathsf{Event}_{\mathsf{hashcoll}}$*).*

Since we required that these aborts do not *occur, it must hold that if* $(\mathsf{I}_1, i, \mathcal{S}_1, P_1, \mathsf{CID}_1) \neq (\mathsf{I}_2, j, \mathcal{S}_2, P_2, \mathsf{CID}_2)$ *then* $\mathsf{post}_i.\mathsf{Hash} \neq \mathsf{post}_j.\mathsf{Hash}$. *This concludes the proof.*