Correctness of the state hash for Top Down ERC998

Lukasz Glen
lukasz.glen@imapp.pl, lukasz.glen@poczta.fm

Each ERC998 bundle can contain ERC20 tokens, ERC721 tokens and other ERC998 bundles. In particular a bundle can contain a bundle. The ERC998 contract prevents from circular dependencies. Thus bundles form a forest: ERC998 bundles as nodes, ERC20, ERC721 and ERC1155 tokens as leaves. A bundle has a state, which is a set of owned tokens and other bundles.

Bundles are valuable objects and they are subjects to trade. So we need a protection from modifying a bundle just before a trade. A kind of state indicator that is easy to be checked, that is changed whenever the state of the bundle is changed or a state of any bundle in its subtree is changed.

A short remark on the scenario where there is a threat of modifying a bundle at trade. The seller makes an offer to sell the bundle, the buyer sends the transaction, just before executing the transaction the seller empties the bundle, the pays for the empty bundle. This is how a state indicator can prevent such fraud. The seller makes an offer to sell the bundle, the offer includes the state indicator, the buyer sends the transaction, if the bundle is modified before executing the transaction then the indicator is changed, at the transaction execution the indicator is checked and the transaction is reverted.

Let $I \in \mathbb{Z}_+$, $\mathbf{P} \subset \mathbb{Z}_+$, $\mathbf{Q} \subset \mathbb{Z}_+$, $\mathbf{R} \subset \mathbb{Z}_+$, such that sets $\{I\}, \mathbf{P}, \mathbf{Q}, \mathbf{R}$ are disjoint. You can think of I as the address of the bundle contract, $\mathbf{P}$ as the addresses of all ERC20 contracts, $\mathbf{Q}$ as the addresses of all ERC721 contracts, $\mathbf{R}$ as the addresses of all ERC1155 contracts. The state S_i of i-th bundle, $i \in \mathbb{Z}_+$, is

$$S_i = \{S_{i,erc20}, S_{i,erc721}, S_{i,erc1155}, S_{i,bundle}\}$$

where

$$S_{i,erc20}(p) \in \mathbb{N} \quad \text{for} \quad p \in \mathbf{P},$$

$$S_{i,erc721}(q) \subset \mathbb{Z}_+ \quad \text{for} \quad q \in \mathbf{Q},$$

$$S_{i,erc1155}(r,t) \subset \mathbb{Z}_+ \quad \text{for} \quad r \in \mathbf{R}, t \in \mathbb{Z}_+,$$

$$S_{i,bundle} \subset \mathbb{Z}_+.$$

you can think of t as tokenId of ERC1155. The initial state is $S_{i,erc20}(p) = 0$, $S_{i,erc721}(q) = \emptyset$, $S_{i,erc1155}(r,t) = 0$, $S_{i,bundle} = \emptyset$.

A state can be changed upon a transaction, a state of one bundle within one transaction. So within one transaction we perform one of the following operations.

1. $addERC20(i,p,v)$ provides a change $S_{i,erc20}(p) := S_{i,erc20}(p) + v$,
2. $removeERC20(i,p,v)$ provides a change $S_{i,erc20}(p) := S_{i,erc20}(p) - v$, assuming that output $S_{i,erc20}(p) \geq 0$,

3. $addERC721(i, q, a)$ provides a change $S_{i,erc721}(q) := S_{i,erc721}(q) \cup \{a\}$, assuming that input $a \notin S_{i,erc721}(q)$,

4. $removeERC721(i, q, a)$ provides a change $S_{i,erc721}(q) := S_{i,erc721}(q) \setminus \{a\}$, assuming that input $a \in S_{i,erc721}(q)$,

5. $addERC1155(i, r, t, v)$ provides a change $S_{i,erc1155}(r, t) := S_{i,erc1155}(r, t) + v$,

6. $removeERC1155(i, r, t, v)$ provides a change $S_{i,erc1155}(r, t) := S_{i,erc1155}(r, t) - v$, assuming that output $S_{i,erc1155}(r, t) \geq 0$,

7. $addNode(i, j)$ provides a change $S_{i,bundle} := S_{i,bundle} \cup \{j\}$, assuming that input $j \notin S_{i,bundle}$ and no circular dependency is created,

8. $removeNode(i, j)$ provides a change $S_{i,bundle} := S_{i,bundle} \setminus \{j\}$, assuming that input $j \in S_{i,bundle}$.

We define the state hash H_i of i-th bundle. It is changed by a transaction. The initial value is

$$H_i = \text{keccak256}(I, i).$$

An operation changes a state hash as follows. Bundles form a forest, so each bundle i has an assigned root bundle $root(i)$. A state hash update is calculated after a state update.

1. $addERC20(i, p, v)$ provides a change $H_{root(i)} := \text{keccak256}(H_{root(i)}, i, p, w)$, where $w = S_{i,erc20}(p)$,

2. $removeERC20(i, p, v)$ provides a change $H_{root(i)} := \text{keccak256}(H_{root(i)}, i, p, w)$, where $w = S_{i,erc20}(p)$,

3. $addERC721(i, q, a)$ provides a change $H_{root(i)} := \text{keccak256}(H_{root(i)}, i, q, a)$,

4. $removeERC721(i, q, a)$ provides a change $H_{root(i)} := \text{keccak256}(H_{root(i)}, i, q, a)$,

5. $addERC1155(i, r, t, v)$ provides a change $H_{root(i)} := \text{keccak256}(H_{root(i)}, i, r, t, w)$, where $w = S_{i,erc1155}(r, t)$,

6. $removeERC1155(i, r, t, v)$ provides a change $H_{root(i)} := \text{keccak256}(H_{root(i)}, i, r, t, w)$, where $w = S_{i,erc1155}(r, t)$,

7. $addNode(i, j)$ provides a change $H_{root(i)} := \text{keccak256}(H_{root(i)}, i, I, H_j)$,

8. $removeNode(i, j)$ provides simultanous changes:

   (a) $H_{root(i)} := \text{keccak256}(H_{root(i)}, i, I, H_j)$,

   (b) $H_j := \text{keccak256}(H_{root(i)}, j, I, H_j)$.

Note that 1 and 2, 3 and 4, 5 and 6, 7 and 8(a) have the same formula. So if a state is changed then a state hash of a root bundle is updated. Note that 7 removes one root bundle, and 8 adds one root bundle.

Our goal is to prove that the state hash is a good indicator of the state in the broader meaning - the state of the bundle and all its descendants. Good means that it is cryptographically hard to find two different states that have the same state hash (a collision). $keccak256$ is not injective. This causes troubles when providing a mathematical proof. We make the following assumption. Note that the state hash is an expression containing nested $keccak256$. We assume that it is cryptographically hard to find a collision for such expressions. Having this we interpret the state hash H as a formal expression (a string of characters) rather than evaluating this expression. Thus we need to prove that two different states cannot be associated with the same state hash. But only for root bundles.

We say that a state hash H is proper if there exists a sequence of transactions that H is a state hash of some bundle.

The depth $D(H)$ is defined as: it is a number of keccak256 in the expression.

If H is the state hash of i-th bundle then we say that $id(H) = i$.

If the i-the bundle is a root then it has $subtree(i)$. More formally, $subtree(i)$ is the smallest set that: $i \in subtreee(i)$, if $j \in subtree(i)$ and $k \in S_{j,bundle}$ then $k \in subtree(i)$.

Fact. Let H be proper.

a) $id(H)$ is well defined (it is a function).

b) If i-th bundle is a root with the state hash H_i, then for every $j \in subtree(i)$ the state S_j is uniquely determined.

Proof. We use mathematical induction.

The statements hold when $D(H) = 1$.

Say that the statements hold for all $\bar{H}$ such that $D(\bar{H}) < N$. Assume that $D(H) = N$.

The state hash H has one of the following forms:

$$H = \text{keccak256}(H', j, a, w),$$

$$H = \text{keccak256}(H', j, a, t, w).$$

Assume that $H = \text{keccak256}(H', j, a, t, w)$. It means that $a \in R$ and, the last operation was either $addERC1155(j, a, t, v)$ or $removeERC1155(j, a, t, v)$, $H' = H'_{root(j)}$. It holds $id(H) = id(H')$ which proves a). Assume that $i = id(H') = root(j)$ and i-th bundle is a root. The state S'_j is uniquely determined and $j \in subtree(i)$. So the last operation is also uniquely determined unless it is either $addERC1155(j, a, t, 0)$ or $removeERC1155(j, a, t, 0)$ but these do not change the state. Thus the state S_j is uniquely determined. b) is proven.

Assume that $H = \text{keccak256}(H', j, a, w)$ and $a \in P$. The prove is the same as above.

Assume that $H = \text{keccak256}(H', j, a, w)$ and $a \in Q$. The prove is the similar as above. The last operation is uniquely determined. It is either

$addERC721(j,a,w)$ or $removeERC721(j,a,w)$. If $w \in S'_{j,erc721}(a)$ then $w \notin S_{j,erc721}(a)$ and the last operation is *remove*. If $w \notin S'_{j,erc721}(a)$ then $w \in S_{j,erc721}(a)$ and the last operation is *add*.

Assume that $H = \text{keccak256}(H', j, I, w)$. Then $w = \bar{H}'$ is some state hash. Note that $D(\bar{H}') < N$. Let $k = id(\bar{H}')$. If $j \neq k$, then the last operation was either $addNode(j,k)$ or $removeNode(j,k)$ but in latter case the hash state transition takes the form 8(a). And we can proceed with the proof as above. So let assume then that $j = k$. The last operation was $removeNode(j,k)$ and the hash state transition was of the form 8(b). So $id(H) = k$ and a) is proven. Note that $subtree(k)$ is the same before and after the operation. After the operation k-th bundle is a root. Also for every $l \in subtree(k)$ it holds $S'_l = S_l$. This proves b) and completes the whole proof. ■

Conclusion.

a) The state hash of a root bundle uniquely determines the state of the whole subtree.

b) The state hash of a root bundle uniquely determines a sequence of transactions that yielded this state hash, with the exception that it does not distinct between $addERC20(i,p,0)$ and $removeERC20(i,p,0)$, $addERC1155(i,r,t,0)$ and $removeERC1155(i,r,t,0)$. In other words, the state hash encodes not only the state but also the history of how this state was created.

c) The state hash is better than last modification time or changes counter. Those do not determine the state and this leaves some space for adversaries.

d) Although it has some limitations. The order of independent changes to children bundles before they were added cannot be determined by the state hash. For instance.

$$addERC20(2,r,1), \quad addERC20(3,r,1), \quad addNode(1,2), \quad addNode(1,3)$$

and

$$addERC20(3,r,1), \quad addERC20(2,r,1), \quad addNode(1,2), \quad addNode(1,3)$$

are sequences of transactions that yield the same H_1. It can be overcome by adding a global transaction counter, if there is enough practical reasoning.