Algorithm 1 The challenger for the burn protocol game-based security.

1: **function** SPEND-ATTACK$_{\mathcal{A},\Pi}(\kappa)$
2: $(t, m, \sigma, pk) \leftarrow \mathcal{A}(1^\kappa)$
3: **return** (BurnVerify$(1^\kappa, t, pk) \wedge$ SpendVerify(m, σ, pk))
4: **end function**

Algorithm 2 The challenger for the burn protocol game-based security.

1: **function** BIND-ATTACK$_{\mathcal{A},\Pi}(\kappa)$
2: $(t, t', \mathsf{burnAddr}) \leftarrow \mathcal{A}(1^\kappa)$
3: **return** $(t \neq t' \wedge$ BurnVerify$(1^\kappa, t, \mathsf{burnAddr}) \wedge$ BurnVerify$(1^\kappa, t', \mathsf{burnAddr}))$
4: **end function**

These two functionalities are typically implemented using a public key signature scheme and accompanied by a respective signing algorithm. The signing algorithm is irrelevant for our burn purposes, as burning entails the inability to spend. As the format of m is cryptocurrency-specific, we intentionally leave it undefined. In both Bitcoin and Ethereum, m corresponds to transaction data. When a new candidate transaction is received from the network, the blockchain node calls SpendVerify, passing the public key pk, which is the address spending money incoming to the new transaction m, together with a signature σ, which signs m and should be produced using the respective secret key.

To state that the protocol generates addresses which cannot be spent from, we introduce a game-based security definition. The unspendability game SPEND-ATTACK is illustrated in Algorithm 1.

Definition 4 (Unspendability). *A burn protocol Π is* unspendable *with respect to a blockchain address protocol Π_α if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\kappa)$ such that* $\Pr[\text{SPEND-ATTACK}_{\mathcal{A},\Pi}(\kappa) = \mathit{true}] \leq \mathsf{negl}(\kappa)$.

It is desired that a burn address encodes one and only one tag. Concretely, given a burn address burnAddr, BurnVerify$(1^\kappa, t,$ burnAddr) should only evaluate to true for a single tag t. The game BIND-ATTACK in Algorithm 2 captures this property.

Definition 5 (Binding). *A burn protocol Π is* binding *if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there is a negligible function $\mathsf{negl}(\kappa)$ such that* $\Pr[\text{BIND-ATTACK}_{\mathcal{A},\Pi}(\kappa)] \leq \mathsf{negl}(\kappa)$.

We note here that the correctness and binding properties of a burn protocol are irrespective of the blockchain address protocol it was designed for.

We are now ready to define what constitutes a *secure proof-of-burn protocol.*

Definition 6 (Security). *Let Π be a correct burn protocol. We say Π is* secure *with respect to a blockchain address protocol Π_α if it is* unspendable *and* binding *with respect to Π_α.*

The aforementioned properties form a good basis for a burn protocol. We observe that it may be possible to detect whether an address is a burn address. While this is desirable in certain circumstances, it allows miners to censor burn transactions. To mitigate this, we propose *uncensorability*, a property which mandates that a burn address is indistinguishable from a regular address if its tag is not known. During the execution of protocols which satisfy this property, when the burn transaction appears on the network, only the user who performed the burn knows that it constitutes a burn transaction prior to revealing the tag. Naturally, as soon as the tag is revealed, *correctness* mandates that the burn transaction becomes verifiable.

Definition 7 (Uncensorability). *Let $\mathcal{T}$ be a distribution of tags. A burn protocol Π is* uncensorable *if the distribution ensembles* $\{(pk, sk) \leftarrow \mathsf{GenAddr}(1^\kappa); pk\}_\kappa$ *and* $\{t \leftarrow \mathcal{T}; pk \leftarrow \mathsf{GenBurnAddr}(1^\kappa, t); pk\}_\kappa$ *are computationally indistinguishable.*

3 Construction

We now present our construction for an uncensorable proof-of-burn protocol. To generate a burn address, the tag t is hashed and a perturbation is performed on the hash by toggling the last bit. Verifying a burn address burnAddr encodes a certain tag t is achieved by invoking GenBurnAddr with tag t and checking whether the result matches burnAddr. If it matches, the burnAddr correctly encodes t. Our construction is illustrated in Algorithm 3.

We outline the blockchain address protocol for Bitcoin Pay to Public Key Hash (P2PKH) [2], with respect to which we prove our construction secure and uncensorable in Section 5. It is parametrized by a secure signature scheme S and a hash function H (for completeness, we give a construction which includes the concrete hash functions and checksums