Our mechanism in principle allows burning from any proof-of-work-based cryptocurrency.

(iv) **Experimental results.** We provide a compehensively tested production grade implementation of the bootstrapping mechanism in Ethereum written in Solidity, which we release as open source software. Our implementation can be used to consume proofs of burn of a source blockchain within a target blockchain. We provide experimental measurements for the cost of burn verification and find that, in current Ethereum prices, burn verification costs $0.28 per transaction. This allows coins burned on one blockchain to be consumed on another for the purposes of, for example, ERC-20 tokens creation [32].

Workflow. A user who wishes to burn her coins generates an address which we call a *burn address*. This address encodes some user-chosen metadata called the *tag*. She then proceeds to send any amount of cryptocurrency to the burn address. After burning her cryptocurrency, she proves to any interested party that she irrevocably destroyed the cryptocurrency in question.

Properties. We define the following properties for a proof-of-burn protocol:

- **Unspendability.** No one can spend the burned cryptocurrency.
- **Binding.** The burn commits only to a single tag.
- **Uncensorability.** Miners who do not agree with the scheme cannot censor burn transactions.

Finally, we consider the *usability* of a proof-of-burn protocol important: whether a user is able to create a burn transaction using her regular cryptocurrency wallet.

Notation. We use $\mathcal{U}(S)$ to denote the uniform distribution obtained by sampling any item of the finite set S with probability $\frac{1}{|S|}$. We denote the support of a distribution $\mathcal{D}$ by $[\mathcal{D}]$. We also use $[n]$ to denote the set of integers from 1 to n. We denote the empty string by ϵ and string concatenation by $\|$.

2 Defining Proof-of-Burn

We now formally define what a proof-of-burn protocol is. Let κ be the security parameter. The protocol consists of two functions GenBurnAddr and BurnVerify and works as follows. Alice first generates an address burnAddr to which she sends some cryptocurrency. The address is generated by invoking GenBurnAddr$(1^\kappa, t)$ and encodes information contained in a tag t, a string of Alice's choice. When the transaction is completed, she gives the transaction and tag to Bob who invokes BurnVerify$(1^\kappa, t,$ burnAddr$)$ to verify she irrevocably destroyed the cryptocurrency while committing to the provided tag.

Definition 1 (Burn protocol). *A* burn *protocol Π consists of two functions* GenBurnAddr$(1^\kappa, t)$ *and* BurnVerify$(1^\kappa, t,$ burnAddr$)$ *which work as follows:*

- GenBurnAddr$(1^\kappa, t)$*: Given a tag $t \in \{0,1\}^*$, generate a* burn address.
- BurnVerify$(1^\kappa, t,$ burnAddr$)$*: Given a tag $t \in \{0,1\}^*$ and an address* burnAddr*, return* true *if and only if* burnAddr *is a burn address and correctly encodes t.*

We require that the burn scheme is *correct*.

Definition 2 (Correctness). *A burn protocol Π is* correct *if for all $t \in \{0,1\}^*$ and for all $\kappa \in \mathbb{N}$ it holds that* BurnVerify$(1^\kappa, t,$ GenBurnAddr$(1^\kappa, t)) =$ true.

With foresight, we remark that the implementation of GenBurnAddr and BurnVerify will typically be deterministic, which alleviates the need for a probabilistic correctness definition.

Naturally, for GenBurnAddr to generate addresses that "look" valid but are unspendable according to the blockchain protocol requires that the burn protocol respects its format. We abstract the address generation and spending verification of the given system into a *blockchain address protocol*:

Definition 3 (Blockchain address protocol). *A* blockchain address protocol *Π_α consists of two functions* GenAddr *and* SpendVerify*:*

- GenAddr(1^κ)*: Returns a tuple* (pk, sk)*, denoting the cryptocurrency address* pk *(a public key) used to receive money and its respective secret key* sk *which allows spending from that address.*
- SpendVerify$(m, \sigma,$ pk$)$*: Returns* true *if the transaction m spending from receiving address* pk *has been authorized by the signature σ (by being signed by the respective private key).*

We note that, while the blockchain address protocol is not part of the burn protocol, the *security* properties of a burn protocol Π will be defined *with respect to* a blockchain address protocol Π_α.