Fourth of the Offslack Specs.

What

We define the notion of one-shot signatures, which are signatures where any secret key can be used to sign only a single message, and then self-destructs. While such signatures are of course impossible classically, we construct one-shot signatures using quantum no-cloning. In particular, we show that such signatures exist relative to a classical oracle, which we can then heuristically obfuscate using known indistinguishability obfuscation schemes. - [

Source](https://eprint.iacr.org/2020/107)

The above source uses the no-cloning theorem, but perhaps we could use TEEs! This would look like the TEE managing a key and only allowing it to be used for a signature once.

How

This could be implemented as a package in the SUAVE standard library.

Desired functionality:

- Generate new one shot pubkey (w/ some source of randomness?)
- Get Generated pubkeys
- Transfer Owner and up keeping ownership map
- Sign message with pubkey
- Verify whether pubkey is used or not

Interfaces:

interface OneShot { function generate() returns (bytes memory); //returns pubkey function sign(bytes pubkey) returns (bytes memory) // returns signature function getPubkeys() reutrns (address[] memmory) function transferOwner(bytes pubkey, address newOwner) reutrns (address[] memmory) }

h/t: @Ferran

Discussion

One open question is how can we let other domains know of the one shot signature keys as they look like regular ECDSA SECP2561k. Perhaps we could publish a merkle trie root of keys? Or maybe we could use a form of hierarchical key derivation to know a public key came from a specific parent key that can only generate TEE based one shot signature keys.

Can we also expand this pattern to one shot programs?