I recently published Cicada, a Solidity library for private on-chain voting (focusing on running tally privacy). Sharing here to open up discussion around potential variants, extensions, and improvements to Cicada. This is the GitHub repo (GitHub - a16z/cicada: A protocol for private on-chain voting, implemented in Solidity.) and accompanying blog post (Building Cicada: Private on-chain voting using time-lock puzzles).

For a description of how Cicada works, please refer to the README in the GitHub repository.

Research directions

1. Extension to >2 choice votes

(multi-candidate election). Currently, Cicada assumes ballot values are binary (0 or 1). You naively extend this to >2 choices by instantiating a binary vote per choice, and proving in zero-knowledge that the sum of the ballot values sum to 1. Section 4.2.3 of Jurik '03 (“Extensions to the Paillier Cryptosystem with

Applications to Cryptological Protocols”) suggests some alternative approaches.

1. Token-weighted voting and other voting systems

. To implement token-weighted voting, you’d need a ZK key-value data structure. I think you could use two instances of Semaphore: a key set (voter address) and value set (voting power) which share a commons set of nullifiers (effectively associating keys with values). Then for Cicada, you’d need to prove that the ballot value is either 0 or the voter’s voting power. Note that exponential ElGamal requires that the final tally be small (< 2^32 ballpark) so you may need to truncate the trailing bits of voting power (arguably you’d want to do so anyway, otherwise the exact final tally values would leak information about how people voted). Supporting other voting systems, e.g. cardinal voting, would be interesting areas of research as well.

1. Coercion-resistance/receipt-freeness

. Coercion-resistance is hard but maybe we could enable a weak form of coercion-resistance by allowing voters to change their ballot while voting is still active, ideally without broadcasting that the vote has been changed.

1. Recursive SNARK composition

. Cicada as currently implemented is already fairly expensive (about 400-450k gas to cast a ballot) for Ethereum mainnet. Implementing the extensions described in the above would likely significantly increase computational costs. We could keep verifier gas cost reasonable by implementing the sigma protocol verifier as a circuit for some generalized proof system, hopefully without too much prover overhead.