Silent Vote (dApp)

User Interface (Frontend)

Anyone with a cryptocurrency wallet and internet connection can interact with the blockchain.

Smart Contract (Backend)

The blockchain contains all information of every transaction.

silentvote.com (Website)

- Connects user with smart contract

bool Vote(candidate)
Midnight Lace Info + candidate + time => Proof A
Smart Contract.vote(Proof)
Returns if vote was successful

int getVotes(candidate)
Midnight Lace Info + candidate + time => Proof B
Smart Contract.getVotes(candidate)
Returns number of votes

bool hasVoted()
Midnight Lace Info => Proof C
Alice sends Proof C to anyone who wishes to verify her civic responsibilities.

Proof Server (Docker)

- Generates a proof of eligibility

bool hasVoted, string name, string candidate, int time => [Proof Server A] => Proof A

Name CANNOT be deduced from the proof alone

- Generates proof of identity

string name, string candidate, int time => [Proof Server B] => Proof B

Only users who are eligible to vote can see results after election results are finalized.

This prevents malicious actors from using timing side channels to correlate votes to voters.

- Generates proof of voter status

string name => [Proof Server C] => Proof C

Alice can send the proof "Alice successfully voted." to Bob and Bob can independently verify this with the public verifier.

Midnight Lace (Chrome Wallet)

- Connects users to dApps
- Stores cryptocurrencies
- Manages digital identities

Verification Algo (Kuchinu)

- Verifies a proof of eligibility

Proof A => [Verification Algo A] => ? Valid
Not Valid: Return false
Valid: SmartContract.vote(candidate)

Proof B => [Verification Algo B] => ? Valid
Not Valid: Return -1
Valid: SmartContract.getVote(candidate)

Proof C => [Verification Algo C] => ? Voted
Valid: Return true
Not Valid: Return false

Voting Algo (snarkJS)

```
bool vote(candidate){
  votesRecieved[candidate]++
  return true
}
```

- All votes are stored anonymously

```
int getVotes(candidate){
  return votesRecieved[candidate]
}
```

- Only certain accounts at certain times can access the data

Verification Server (snarkJS)

Proof D => [Verification Algo D] => Valid: Results are accurate. Not Valid: Results have been tampered!!!

Proving Server (Docker)

- Generates proof of results

SmartContract.votesRecieved[candidate] => [Proof Server D] => Proof D

Results will always be published with the proof so anyone can verify it.

Proof A verifies the user is eligible to vote and election is running.

Proof B verifies the user and time are eligible to check results.

Proof C verifies that a user has voted successfully.

User proves, Server verifies

Proof D verifies the results are what users voted for.

Server proves, user verifies

Users		Shielded Ledger [A,B,C,D]	Unshielded Ledger	
Alice (Admin)	vote(Bob)	0,1,0,0	Vote Transaction Successful	HIDDEN
Bob	vote(Bob)	0,2,0,0	Vote Transaction Successful	
Carol	vote(David)	0,2,0,1	Vote Transaction Successful	
David	vote(Alice)	1,2,0,1	Vote Transaction Successful	
David	vote(Alice)	1,2,0,1	Vote Transaction Unsuccessful	
Alice (Admin)	getVotes(A)	1	Admin Accessed Votes	PUBLIC
	getVotes(B)	2	Admin Accessed Votes	
	getVotes(C)	0	Admin Accessed Votes	
	getVotes(D)	1	Admin Accessed Votes	

These 4 proofs effectively make elections more secure, anonymous, and trustworthy than physical ballot voting. Voter data is only ever stored on local machines so no data can be stolen in a data breach. All voters can be assured that election results aren't rigged or miscounted if they can verify it themselves. Even a spy eavesdropping on the connection will gain nothing. There is even an added feature that someone can prove they voted, something that is impossible to do with physical ballots.

Silent Vote and other blockchains with zero trust architecture are the future of all online transactions!