Eco Take-home Challenge

Challenge Requirements

Initial Thoughts

We have 2 parties, they need to sign a secret off-chain, push it on-chain on a specific block, and then have the ability to reveal it later on. This immediately brings EIP-712 to mind, we don't necessarily know the form of the data, but even just a signed blob of bytes would benefit from the extra details EIP-712 gives us and we can attach some signer information. This could also probably be done with ERC-191 but the domain logic is useful, and presumably we could extend the reveal function to support additional payload types without needing to modify the commit function. EIP-712 is usually used for single-signer situation so some extra considerations need to be taken into account. The only unknown is *what* exactly are they signing. It's not fully clear from the challenge whether all the data explicitly needs to exist on-chain, so the lightest weight solution is going to be signing a hash, pushing that on chain, and then verifying the secret hashes into this signed value in the "reveal" call. Alternatively we could try and push the full data on-chain using address as a public key, but this doesn't really expand the functionality and massively increases the complexity as we've need two encrypted payloads, as well as two signatures.

Challenges

- We want a single block, so one party has to have all of the information that's going on chain
  - Probably best to store the counter-party's signature on IPFS or something, out of scope for this challenge though.
- Both parties need to have signed off on the data
  - We'll be using ECDSA and EIP-712 for this
- Both parties need to be able to reveal the data

- We’ll have both parties provide signatures for the hash, the hash will be a typed structure data with the two signing parties and a bytes payload

Proposed Solution

We’re only going to need 2 endpoints, a commit endpoint, and a reveal endpoint.
The commit endpoint will look something like:

```
function commit(bytes32 secretHash, uint8 v1, uint8 v2, bytes32 r1, bytes32 r2, bytes32 s
1, bytes32 s2) public
```

with our reveal endpoint looking something like:

```
function reveal(address signerOne, address signerTwo, bytes secret) public
```

We provide both signers mainly to avoid needing to deduce ordering for the signed data.

We’ll also need a secret structure:

```
struct Secret {
  address signerOne;
  address signerTwo;
  bytes secret;
}
```

When the secret is revealed, we confirm that the signatures match up to the listed signers, that both signatures correspond to the provided secret, and that the revealing party is one of the two initial signers.

Assumptions

- I’m assuming we only want one of the two parties to be able to reveal the data, so I’m checking against the signer to ensure they’re one of the two signing parties. It would be simple to just omit that check if that assumptions not valid.

- In all cases where it was relevant I optimized for computation vs storage
- I used raw string require statements assuming that readability and simplicity was best in this case, I'm aware that using

Additional Notes

- I use https://gitmoji.dev/ for all the commits - commit purpose adheres as close as possible to this standard.
- These notes were taken in Notion during ideation, easily could have been done directly in the repo's readme but I'm more comfortable keeping them separate.
- Initial proposed solution interface changed slightly through implementation process.
- In one of the last commits (`67d429b`) I switched from string based requires to custom error based revert. I don't have a super strong opinion which is better, it really depends on the situation so I provided both. In this case custom errors did lead to some small gas savings:

```
test_HashStruct() (gas: 0 (0.000%))
testFuzz_Commit(bytes) (gas: 9 (0.005%))
test_Commit() (gas: 9 (0.005%))
test_Reveal() (gas: 12 (0.019%))
test_RevertIf_InvalidSender() (gas: -55 (-0.029%))
test_RevertIf_InvalidCounterparty() (gas: -72 (-0.036%))
test_RevertIf_CommitExists() (gas: -75 (-0.040%))
test_RevertIf_InvalidSender() (gas: -75 (-0.166%))
testFuzz_RevertIf_CommitDoesNotExist(string) (gas: -87 (-0.488%))
test_RevertIf_CommitDoesNotExist() (gas: -87 (-0.519%))
Overall gas change: -421 (-0.038%)
```

- I tried to make every commit as small and atomic as possible, I didn't do any PRs either, but in a shared repo this would have been spread across multiple PRs just didn't simulate that for this exercise.
- The repo is posted privately to my GitHub, I'd be happy to invite anyone to that if it's easier to review that way, just get me a username to invite.