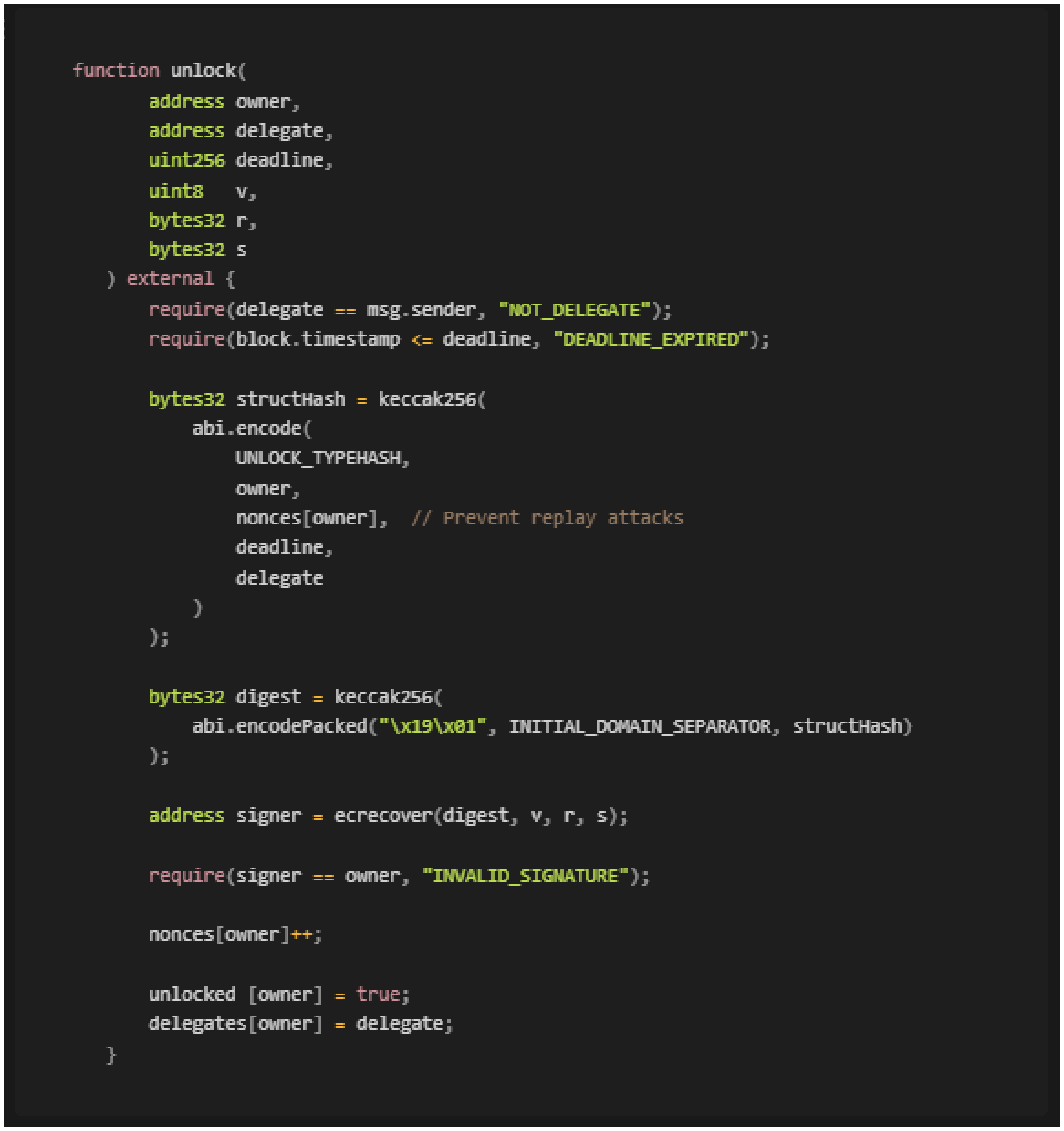
Unlock Mechanism Audit for contracts/src/Manager.sol



```solidity
function unlock(
        address owner,
        address delegate,
        uint256 deadline,
        uint8   v,
        bytes32 r,
        bytes32 s
    ) external {
        require(delegate == msg.sender, "NOT_DELEGATE");
        require(block.timestamp <= deadline, "DEADLINE_EXPIRED");

        bytes32 structHash = keccak256(
            abi.encode(
                UNLOCK_TYPEHASH,
                owner,
                nonces[owner],  // Prevent replay attacks
                deadline,
                delegate
            )
        );

        bytes32 digest = keccak256(
            abi.encodePacked("\x19\x01", INITIAL_DOMAIN_SEPARATOR, structHash)
        );

        address signer = ecrecover(digest, v, r, s);

        require(signer == owner, "INVALID_SIGNATURE");

        nonces[owner]++;

        unlocked [owner] = true;
        delegates[owner] = delegate;
    }
```

[H1] - Signature can be replayed on another Chain.

The contract will be deployed on Arbitrum and Optimism. `owner`, `nonce`, `deadline`, and `delegate` can be true on two chains simultaneously. The hash should include chain ID. It didn't. It has one UNLOCK_TYPEHASH which has not been defined somewhere in the contract.

The fix:

Simply hardcode the `chainId` for each deployment and include it in the structHash.

[M1] - Given that `deadline` is intended to be the time to revoke the unlock, the current implementation does not support the functionality.

`require(block.timestamp <= deadline, "DEADLINE_EXPIRED";`
currently checks if the current block time is before or equal to the deadline. If it is after, the function would revert, implying the action cannot be performed now.

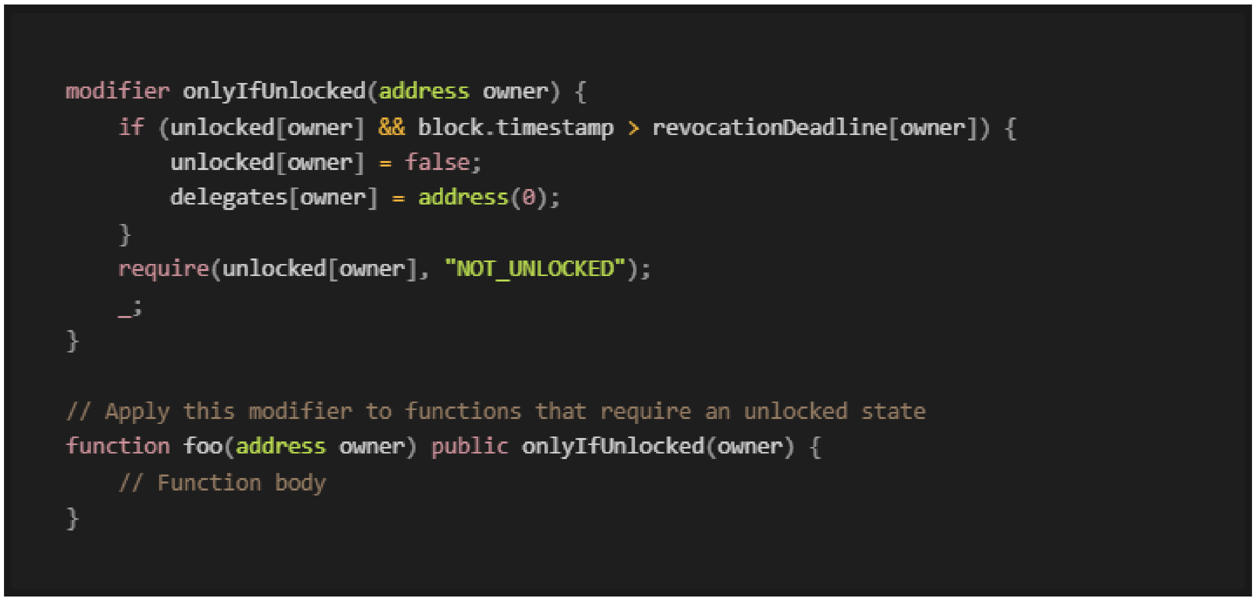
Look again into what this deadline is doing in the function.

Description

You don't necessarily need the `deadline` in the lock function if the goal is to automate or simplify the revocation process. Here are two alternative approaches to manage the revocation without explicitly checking the deadline in the lock function:

1. Automatic Revocation Check in Every Relevant Function

Instead of having the lock function check the deadline, you can ensure that every function requiring an unlocked state checks if the deadline has passed and automatically revokes if necessary. This would eliminate the need for a separate lock function or for passing the deadline:

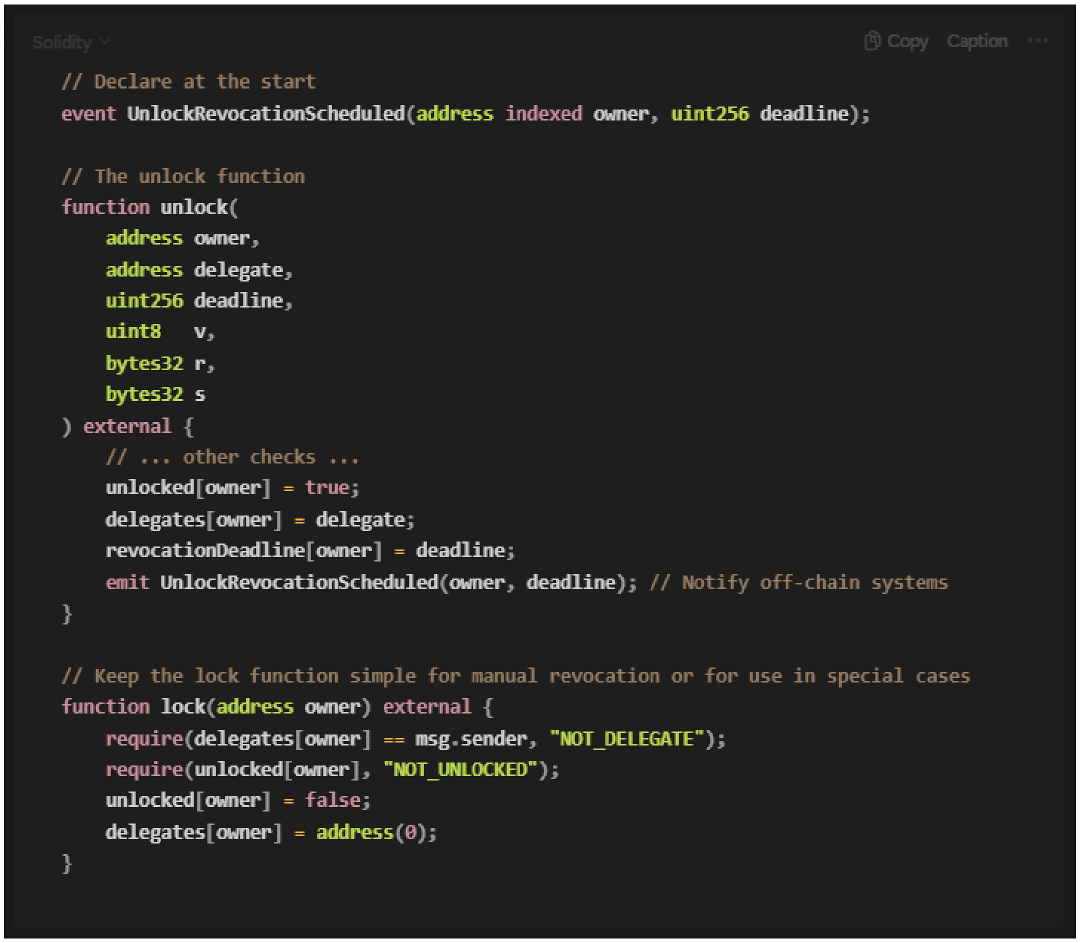


```
modifier onlyIfUnlocked(address owner) {
    if (unlocked[owner] && block.timestamp > revocationDeadline[owner]) {
        unlocked[owner] = false;
        delegates[owner] = address(0);
    }
    require(unlocked[owner], "NOT_UNLOCKED");
    _;
}

// Apply this modifier to functions that require an unlocked state
function foo(address owner) public onlyIfUnlocked(owner) {
    // Function body
}
```

But you must ensure all relevant functions use this modifier.

1. **Use Event for Off-Chain Revocation Management**



```
// Declare at the start
event UnlockRevocationScheduled(address indexed owner, uint256 deadline);

// The unlock function
function unlock(
    address owner,
    address delegate,
    uint256 deadline,
    uint8   v,
    bytes32 r,
    bytes32 s
) external {
    // ... other checks ...
    unlocked[owner] = true;
    delegates[owner] = delegate;
    revocationDeadline[owner] = deadline;
    emit UnlockRevocationScheduled(owner, deadline); // Notify off-chain systems
}

// Keep the lock function simple for manual revocation or for use in special cases
function lock(address owner) external {
    require(delegates[owner] == msg.sender, "NOT_DELEGATE");
    require(unlocked[owner], "NOT_UNLOCKED");
    unlocked[owner] = false;
    delegates[owner] = address(0);
}
```

• On-chain gas savings as revocation logic moves off-chain, but benefits of managing the things onchain are lost.

[mi](http://x.com/designer_misbah)