AKIRA TECH

PROJECT

Ʉnlock Protocol

CLIENT

Ʉnlock

DATE

January 2022

REVIEWERS

Andrei Simion

@andreiashu

Table of Contents

Details

- **Client** Ʉnlock
- **Date** January 2022
- **Reviewers** Andrei Simion (@andreiashu)
- **Repository:** Ʉnlock Protocol
- **Commit hash** `b709f19aa3217202caa3414247222850355b1dbb`

- **Technologies**
    - Solidity
    - Node.JS

Issues Summary

SEVERITY	OPEN	CLOSED
Informational	0	0
Minor	4	0
Medium	2	0
Major	0	0

Executive summary

This report represents the results of the engagement with **Ʉnlock** to review **Ʉnlock Protocol**.

The review was conducted over the course of **1 week** from **January 24th to January 28th, 2022**. A total of **5 person-days** were spent reviewing the code.

The design of the Unlock Protocol platform is thoroughly thought out. Furthermore, the team tried to reach a high level of resilience and decentralization (more on this in the Trust Model section).

A user wanting to accept payments from their subscribers can use Unlock Protocol to deploy a `PublicLock`. This contract acts as a membership gateway for users who wish to access paid content.

The `Unlock` contract is responsible mainly for the deployment and upgrade of Public Locks but otherwise is not essential for a Public Lock to run correctly or accept payments for memberships. In this respect, the team took precautions to ensure that if the Unlock contract gets compromised or suffers a malfunction, the deployed locks are not affected:

```
try unlockProtocol.recordKeyPurchase(inMemoryKeyPrice, _referrer)
{}
catch {
  // emit missing unlock
  emit UnlockCallFailed(address(this), address(unlockProtocol));
}
```

> code extract from `PublicLockV9.sol` - the latest stable version of Public Lock template implementation at the time of writing. If there is an issue in the main `Unlock.sol` contract, the `purchase()` function will continue to function as expected.

The current Public Lock code does not protect against an out of gas attack in the Unlock contract. This issue I identified and raised as part of this review.

Review process

At the beginning of the week, I spent time getting more familiar with the code and the protocol's design.

I started going through the `ERC20Patched.sol` file. For historical reasons the team had to flatten the code for their ERC20 contract code - most of the code is OpenZeppelin libraries apart from some changes to the memory layout.

The bulk of the time was spent on the `Unlock.sol` contract. Users interact with this contract when deploying a new `PublicLock` and upgrade to different versions of a Lock template implementation.

Towards the end of the week, I continued to review the code while creating an overview of the architecture to help me better understand the whole trust model.

Scope

The initial review focused on the Ʉnlock Protocol repository, identified by the commit hash `b709f19aa3217202caa3414247222850355b1dbb`.

I focused on manually reviewing the codebase, searching for security issues such as, but not limited to, re-entrancy problems, transaction ordering, block timestamp dependency, exception handling, call stack depth limitation, integer overflow/underflow, self-destructible contracts, unsecured balance, use of origin, costly gas patterns, architectural problems, code readability.

Includes:

- code/smart-contracts/contracts/ERC20Patched.sol
- code/smart-contracts/contracts/UnlockDiscountTokenV2.sol
- code/smart-contracts/contracts/UnlockDiscountTokenV3.sol
- code/smart-contracts/contracts/Unlock.sol
- code/smart-contracts/contracts/utils/UnlockInitializable.sol
- code/smart-contracts/contracts/utils/UnlockOwnable.sol

Trust Model

Lock owners can run their locks without reliance on the Unlock Protocol

Lock managers can receive payments even with a compromised or malfunctioning Unlock Protocol (`Unlock.sol`) contract. I raised an issue about an edge case whereby a malicious takeover of `Unlock.sol` contract can perform an out of gas attack on Public Locks. That being said, in general, publick locks continue to receive payments regardless of the state of Unlock Protocol.

Privileged Roles and Ownership

A the time of writing, the main `Unlock.sol` contract in the system is controlled by the team's multi-sig. This is done for ease of use and the ability to respond quickly to incidents. The team plans to move to a DAO model of governance once the system becomes more stable and battle tested.

This is a common strategy in DeFi projects: a team will start with a multi-stig that gives flexibility, ease of use, and quick response to incidents. Then, as the project matures, it transitions to a DAO model, a more decentralized way of running it.

Recommendations

I identified a few possible general improvements that are not security issues during the review, which will bring value to the developers and the community reviewing and using the product.

Increase the number of tests

A good rule of thumb is to have 100% test coverage. This does not guarantee the lack of security problems, but it means that the desired functionality behaves as intended. The negative tests also bring value because not allowing some actions to happen is also part of the desired behavior.

Issues

A compromised `Unlock.sol` contract can cause `PublicLockV9.purchase()` to fail through an out of gas attack (or bug)

Status Open Severity Medium

Description

A user purchase is performed through the `purchase` function within a `PublicLock` contract (the latest version at the time of writing being `PublicLockV9`):

code/packages/contracts/src/contracts/PublicLock/PublicLockV9.sol#L3172-L3181

```
function purchase(
  uint256 _value,
  address _recipient,
  address _referrer,
  address _keyManager,
  bytes calldata _data
) external payable
  onlyIfAlive
  notSoldOut
{
```

This function calls the `recordKeyPurchase` on the `Unlock` contract which tallies the sales for the Public Lock but also distributes `UDT` token bonuses:

code/smart-contracts/contracts/Unlock.sol#L323-L337

```
/**
 * This function keeps track of the added GDP, as well as grants of discount tokens
 * to the referrer, if applicable.
 * The number of discount tokens granted is based on the value of the referal,
 * the current growth rate and the lock's discount token distribution rate
 * This function is invoked by a previously deployed lock only.
 * TODO: actually implement
 */
function recordKeyPurchase(
  uint _value,
  address _referrer
)
  public
  onlyFromDeployedLock()
{
```

To make the `purchase()` function more resilient and not depend on the well functioning of the `Unlock` contract, the team uses a `try / catch` block to call the `Unlock.recordKeyPurchase()`. This is a good approach and provides deployed

PublicLock owners additional safety from a compromised or malfunctioning `Unlock` contract:

code/packages/contracts/src/contracts/PublicLock/PublicLockV9.sol#L3238-L3243

```
    try unlockProtocol.recordKeyPurchase(inMemoryKeyPrice, _referrer)
    {}
    catch {
      // emit missing unlock
      emit UnlockCallFailed(address(this), address(unlockProtocol));
    }
```

The issue however is that a malicious actor having access to the `Unlock` contract can cause the `purchase` function to run out of gas by implementing a computationally gas expensive loop within the `Unlock.recordKeyPurchase()` function.

Recommendation

Update the call to `recordKeyPurchase()` to limit the amount of gas the called function can use and thus limit the impact an attack on Unlock can have on already deployed locks.

Lack of validation for `addLockTemplate` can break Locks upgrades; documentation out of sync with code;

Status Open Severity Medium

Description

`addLockTemplate` is used by the owner of the protocol to:

- add new PublicLock template implementations mapped to specific versions
- update existing PublicLock template implementations to new versions

code/smart-contracts/contracts/Unlock.sol#L182-L190

```
  /**
  * @dev Registers a new PublicLock template immplementation
  * The template is identified by a version number
  * Once registered, the template can be used to upgrade an existing Lock
  */
  function addLockTemplate(address impl, uint16 version) public onlyOwner {
    _publicLockVersions[impl] = version;
    _publicLockImpls[version] = impl;
    if (publicLockLatestVersion < version) publicLockLatestVersion = version;
```

There are several issues with the way the code and documentation are written:

- lack of argument checks means that a template implementation can be set to version `0`. If a Lock will try to upgrade to that specific implementation, the operation will fail because `upgradeLock` considers version `0` as invalid:

code/smart-contracts/contracts/Unlock.sol#L283-L285

```
    // make our upgrade
    address impl = _publicLockImpls[version];
    require(impl != address(0), "this version number has no corresponding lock template");
```

- lack of argument checks means that a template implementation can be set to a non-sequential one. If a Lock will try to upgrade to that specific implementation, the operation will fail because `upgradeLock` only allows upgrades to higher, sequential version values:

code/smart-contracts/contracts/Unlock.sol#L278-L281

```
    // check version
    IPublicLock lock = IPublicLock(lockAddress);
    uint16 currentVersion = lock.publicLockVersion();
    require( version == currentVersion + 1, 'version error: only +1 increments are allowed');
```

- the documentation is outdated and doesn't fully cover the whole functionality that the code provides (ie. the updating part):

code/smart-contracts/contracts/Unlock.sol#L183-L185

```
  * @dev Registers a new PublicLock template immplementation
  * The template is identified by a version number
  * Once registered, the template can be used to upgrade an existing Lock
```

Recommendation

My recommendation here is to add sanity checks to this owner-operated function.

In terms of gas costs, I outline below an example whereby simply adding a `require` statement that would fix one of the issues outlined above, will add 29 gas to the function execution costs:

```
// SPDX-License-Identifier: MIT

pragma solidity 0.8.11;


contract Scratch0 {

    mapping(uint16 => address) private _publicLockImpls;
    mapping(address => uint16) private _publicLockVersions;
    // gas 66686
    function config(address impl, uint16 version) public {
        _publicLockVersions[impl] = version;
        _publicLockImpls[version] = impl;
    }
}

contract Scratch1 {

    mapping(uint16 => address) private _publicLockImpls;
    mapping(address => uint16) private _publicLockVersions;


    // gas 66715
    function config(address impl, uint16 version) public {
        require(version > 0);
        _publicLockVersions[impl] = version;
        _publicLockImpls[version] = impl;
    }
}
```

Public Locks can use the `upgradeLock` even if they are not deployed through the Unlock Protocol

Status Open Severity Minor

Description

`upgradeLock` function can be used by `PublicLock` managers to upgrade to the next version of the protocol:

code/smart-contracts/contracts/Unlock.sol#L267-L276

```
  /**
   * @dev Upgrade a Lock template implementation
   * @param lockAddress the address of the lock to be upgraded
   * @param version the version number of the template
   */
  function upgradeLock(address payable lockAddress, uint16 version) external returns(address) {
    require(proxyAdminAddress != address(0), "proxyAdmin is not set");

    // check perms
    require(_isLockManager(lockAddress, msg.sender) == true, "caller is not a manager of this lock");
```

Currently, there is no check to ensure that the lock being upgraded has been originally deployed through the `Unlock` contract.

Recommendation

Use the `onlyFromDeployedLock` modifier to only allow locks deployed with the Unlock Protocol to use the upgrade function:

code/smart-contracts/contracts/Unlock.sol#L58-L61

```
modifier onlyFromDeployedLock() {
  require(locks[msg.sender].deployed, 'ONLY_LOCKS');
  _;
}
```

`configUnlock` does not validate arguments; can lead to incorrect accounting in other parts of the system

Status Open Severity Minor

Description

`configUnlock` function can be used by the multisig wallet or DAO address to update the config values of the protocol:

code/smart-contracts/contracts/Unlock.sol#L450-L462

```
function configUnlock(
  address _udt,
  address _weth,
  uint _estimatedGasForPurchase,
  string calldata _symbol,
  string calldata _URI,
  uint _chainId
) external
  onlyOwner
{
  udt = _udt;
  weth = _weth;
  estimatedGasForPurchase = _estimatedGasForPurchase;
```

The issue however is that there are no sanity checks on the new values. There are several side effects of the owner setting unwanted config values due to human error.

If `weth` is set to a nil address the `grossNetworkProduct` state var will not account for the correct `_value` purchased in the `recordKeyPurchase` function:

code/smart-contracts/contracts/Unlock.sol#L341-L355

```
    if(tokenAddress != address(0) && tokenAddress != weth) {
      // If priced in an ERC-20 token, find the supported uniswap oracle
      IUniswapOracle oracle = uniswapOracles[tokenAddress];
      if(address(oracle) != address(0)) {
        valueInETH = oracle.updateAndConsult(tokenAddress, _value, weth);
      }
    }
    else {
      // If priced in ETH (or value is 0), no conversion is required
      valueInETH = _value;
    }

    grossNetworkProduct = grossNetworkProduct + valueInETH;
    // If GNP does not overflow, the lock totalSales should be safe
    locks[msg.sender].totalSales += valueInETH;
```

Other values that I investigated but are actually left without validation on purpose:

- `udt`, and `estimatedGasForPurchase` can both be set to their nil value (nil address and zero) by the team in order to disable the distribution of tokens on different chains;
- if the `chainId` is set to `0` the wrong number of tokens will be distributed on L2 chains. This variable is left unchecked on purpose by the team because during tests one wants to be able to set the chain to 0 to set the system to behave as on mainnet.

Recommendation

Implement sanity checks as much as possible on the values passed to `configUnlock`.

Division by zero in `recordKeyPurchase` when `grossNetworkProduct` is 0

Status Open Severity Minor

Description

`recordKeyPurchase` is called from locks in order to update specific accounting-related variables:

code/smart-contracts/contracts/Unlock.sol#L353

```
    grossNetworkProduct = grossNetworkProduct + valueInETH;
```

but also to distribute or mint UDT tokens to lock owners:

code/smart-contracts/contracts/Unlock.sol#L409-L416

```
            // Only distribute if there are enough tokens
            IMintableERC20(udt).transfer(_referrer, tokensToDistribute - devReward);
            IMintableERC20(udt).transfer(owner(), devReward);
          }
        } else {
          // No distribnution
          IMintableERC20(udt).mint(_referrer, tokensToDistribute - devReward);
          IMintableERC20(udt).mint(owner(), devReward);
```

In order to determine the number of tokes to transfer to lock owners, there are 2 formulas that are used, depending on which chain the protocol is, that involves dividing by the `grossNetworkProduct`:

code/smart-contracts/contracts/Unlock.sol#L389-L392

```
          maxTokens = IMintableERC20(udt).balanceOf(address(this)) * valueInETH / (2 + 2 * valueInETH /
        } else {
          // Mainnet: we mint new token using log curve
          maxTokens = IMintableERC20(udt).totalSupply() * valueInETH / 2 / grossNetworkProduct;
```

The `grossNetworkProduct` can be re-set to `0` by the owner:

code/smart-contracts/contracts/Unlock.sol#L510-L518

```
  // Allows the owner to change the value tracking variables as needed.
  function resetTrackedValue(
    uint _grossNetworkProduct,
    uint _totalDiscountGranted
  ) external
    onlyOwner
  {
    grossNetworkProduct = _grossNetworkProduct;
    totalDiscountGranted = _totalDiscountGranted;
```

Either because the owner reset the value of `grossNetworkProduct` or the protocol has yet to have any purchases recorded when its value is 0, the calculation of `maxTokens` will throw because of a division by zero error:

code/smart-contracts/contracts/Unlock.sol#L389-L392

```
          maxTokens = IMintableERC20(udt).balanceOf(address(this)) * valueInETH / (2 + 2 * valueInETH /
        } else {
          // Mainnet: we mint new token using log curve
          maxTokens = IMintableERC20(udt).totalSupply() * valueInETH / 2 / grossNetworkProduct;
```

This can happen when the above condition is met and when `valueInEth` is also 0 for non-ETH and non-WETH tokens that do not have an oracle:

code/smart-contracts/contracts/Unlock.sol#L341-L353

```
    if(tokenAddress != address(0) && tokenAddress != weth) {
      // If priced in an ERC-20 token, find the supported uniswap oracle
      IUniswapOracle oracle = uniswapOracles[tokenAddress];
      if(address(oracle) != address(0)) {
        valueInETH = oracle.updateAndConsult(tokenAddress, _value, weth);
      }
    }
    else {
      // If priced in ETH (or value is 0), no conversion is required
      valueInETH = _value;
    }

    grossNetworkProduct = grossNetworkProduct + valueInETH;
```

Recommendation

Check for 0 value `grossNetworkProduct` before performing the calculations in order to avoid a revert in this function.

Documentation typos; code minor fixes;

Status Open Severity Minor

Description

- Documentation typo `nimbers` -> `numbers`

code/smart-contracts/contracts/Unlock.sol#L201

```
  * @param _maxNumberOfKeys the maximum nimbers of keys to be edited
```

- Documentation type `dicount` -> `discount`:

code/smart-contracts/contracts/Unlock.sol#L9

```
 *  2. Grant dicount tokens to users making referrals and/or publishers granting discounts.
```

- Remove obsolete comment: https://github.com/akiratechhq/review-unlock-protocol-2022-01/blob/9ad4bee14829f665e05598da4124eb9a92831e94/code/smart-contracts/contracts/Unlock.sol#L329
- Parenthesis can be omitted for modifiers that do not require arguments:

code/smart-contracts/contracts/Unlock.sol#L141-L146

```
  function initialize(
    address _unlockOwner
  )
    public
    initializer()
  {
```

code/smart-contracts/contracts/Unlock.sol#L331-L336

```
  function recordKeyPurchase(
    uint _value,
    address _referrer
  )
    public
    onlyFromDeployedLock()
```

code/smart-contracts/contracts/Unlock.sol#L428-L434

```
  function recordConsumedDiscount(
    uint /* _discount */,
    uint /* _tokens */
  )
    public
    view
    onlyFromDeployedLock()
```

- In `recordKeyPurchase` the condition for checking the value of the purchase contains about 80 lines of code:

code/smart-contracts/contracts/Unlock.sol#L338-L343

```
    if(_value > 0) {
      uint valueInETH;
      address tokenAddress = IPublicLock(msg.sender).tokenAddress();
      if(tokenAddress != address(0) && tokenAddress != weth) {
        // If priced in an ERC-20 token, find the supported uniswap oracle
        IUniswapOracle oracle = uniswapOracles[tokenAddress];
```

This can be rewritten to make the code easier to follow but returning early:

```
if(_value == 0) {
  return;
}

... rest of the code that relies on a non-zero `_value`
```

- the documentation seems to be outdated since in this case, the `_value` cannot be `0`:

code/smart-contracts/contracts/Unlock.sol#L348-L350

```
    else {
      // If priced in ETH (or value is 0), no conversion is required
      valueInETH = _value;
```

`_value` is checked here:

code/smart-contracts/contracts/Unlock.sol#L338

```
    if(_value > 0) {
```

Artifacts

Surya

Sūrya is a utility tool for smart contract systems. It provides a number of visual outputs and information about the structure of smart contracts. It also supports querying the function call graph in multiple ways to aid in the manual inspection and control flow analysis of contracts.

```
surya mdreport report.md code/smart-contracts/contracts/{Unlock.sol,ERC20Patched.sol,UnlockDiscountTokenV2
```

Files Description Table

File Name	SHA-1 Hash
code/smart-contracts/contracts/Unlock.sol	123c0f2cb193afd49b5c578aee3519
code/smart-contracts/contracts/ERC20Patched.sol	dc3af7e8b18dd7ea963dead6f0fcc3
code/smart-contracts/contracts/UnlockDiscountTokenV2.sol	ff11e9f7237055b5e9f1bb7d0657d84
code/smart-contracts/contracts/UnlockDiscountTokenV3.sol	32397390ca7a4e6c549d5caaff1d97

Contracts Description Table

Contract	**Type**	
└	**Function Name**	**V**
Unlock	Implementation	Unloc Unlo
└	initialize	P
└	initializeProxyAdmin	P
└	_deployProxyAdmin	P
└	publicLockVersions	Ex
└	publicLockImpls	Ex
└	addLockTemplate	P
└	createLock	P
└	createUpgradeableLock	P
└	upgradeLock	Ex
└	_isLockManager	P
└	computeAvailableDiscountFor	P
└	networkBaseFee	Ex
└	recordKeyPurchase	P
└	recordConsumedDiscount	P
└	unlockVersion	Ex
└	configUnlock	Ex
└	setLockTemplate	Ex
└	setOracle	Ex
└	resetTrackedValue	Ex
└	getGlobalBaseTokenURI	Ex
└	getGlobalTokenSymbol	Ex
Roles	Library	
└	add	In
└	remove	In
└	has	In

Contract	Type	
IERC20PermitUpgradeable	Interface	
└	permit	Ex
└	nonces	Ex
└	DOMAIN_SEPARATOR	Ex
IERC20Upgradeable	Interface	
└	totalSupply	Ex
└	balanceOf	Ex
└	transfer	Ex
└	allowance	Ex
└	approve	Ex
└	transferFrom	Ex
IERC20MetadataUpgradeable	Interface	IERC2
└	name	Ex
└	symbol	Ex
└	decimals	Ex
Initializable	Implementation	
ContextUpgradeable	Implementation	In
└	__Context_init	In
└	__Context_init_unchained	In
└	_msgSender	In
└	_msgData	In
ERC20Upgradeable	Implementation	Ini Contex IERC2 IERC20Met
└	__ERC20_init	In
└	__ERC20_init_unchained	In
└	decimals	P

Contract	Type	
└	totalSupply	F
└	balanceOf	F
└	transfer	F
└	allowance	F
└	approve	F
└	transferFrom	F
└	increaseAllowance	F
└	decreaseAllowance	F
└	_transfer	In
└	_mint	In
└	_burn	In
└	_approve	In
└	_beforeTokenTransfer	In
└	_afterTokenTransfer	In
ECDSAUpgradeable	Library	
└	recover	In
└	recover	In
└	recover	In
└	toEthSignedMessageHash	In
└	toTypedDataHash	In
EIP712Upgradeable	Implementation	In
└	__EIP712_init	In
└	__EIP712_init_unchained	In
└	__EIP712_init_unsafe	In
└	_domainSeparatorV4	In
└	_buildDomainSeparator	P
└	_hashTypedDataV4	In
└	_EIP712NameHash	In

Contract	Type	
└	_EIP712VersionHash	In
CountersUpgradeable	Library	
└	current	In
└	increment	In
└	decrement	In
└	reset	In
ERC20PermitUpgradeable	Implementation	Ini ERC20 IERC20Pe EIP712
└	__ERC20Permit_init	In
└	__ERC20Permit_init_unchained	In
└	__ERC20Permit_init_unsafe	In
└	permit	F
└	nonces	F
└	DOMAIN_SEPARATOR	Ex
└	_useNonce	In
MathUpgradeable	Library	
└	max	In
└	min	In
└	average	In
└	ceilDiv	In
SafeCastUpgradeable	Library	
└	toUint224	In
└	toUint128	In
└	toUint96	In
└	toUint64	In
└	toUint32	In

Contract	Type	
└	toUint16	In
└	toUint8	In
└	toUint256	In
└	toInt128	In
└	toInt64	In
└	toInt32	In
└	toInt16	In
└	toInt8	In
└	toInt256	In
ERC20VotesUpgradeable	Implementation	Ini ERC20Pe
└	__ERC20Votes_init_unchained	In
└	__ERC20Votes_init_unsafe	In
└	checkpoints	P
└	numCheckpoints	P
└	delegates	P
└	getVotes	P
└	getPastVotes	P
└	getPastTotalSupply	P
└	_checkpointsLookup	P
└	delegate	P
└	delegateBySig	P
└	_maxSupply	In
└	_mint	In
└	_burn	In
└	_afterTokenTransfer	In
└	_delegate	In
└	_moveVotingPower	P
└	_writeCheckpoint	P

Contract	Type	
└	_add	P
└	_subtract	P
ERC20VotesCompUpgradeable	Implementation	Ini ERC20Vc
└	__ERC20VotesComp_init_unchained	In
└	getCurrentVotes	Ex
└	getPriorVotes	Ex
└	_maxSupply	In
MinterRoleUpgradeable	Implementation	Ini Contex
└	initialize	F
└	isMinter	F
└	addMinter	F
└	renounceMinter	F
└	_addMinter	In
└	_removeMinter	In
ERC20DetailedUpgradeable	Implementation	Ini IERC2(
└	initialize	F
└	name	F
└	symbol	F
└	decimals	F
ERC20MintableUpgradeable	Implementation	Ini ERC20 MinterRc
└	initialize	F
└	mint	F

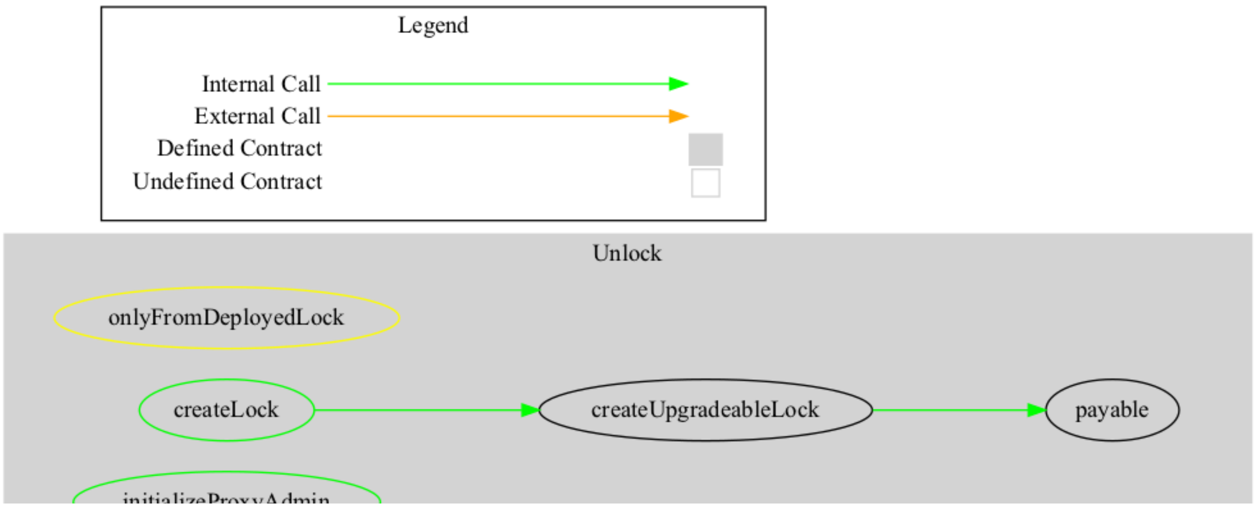
Contract	Type	
UnlockDiscountTokenV2	Implementation	ERC20Mint ERC20Det ERC20Votes
└	initialize	F
└	initialize2	F
└	name	F
└	symbol	F
└	decimals	F
└	_mint	In
└	_burn	In
└	_afterTokenTransfer	In
UnlockDiscountTokenV3	Implementation	UnlockD
└	_beforeTokenTransfer	In
└	_transfer	In

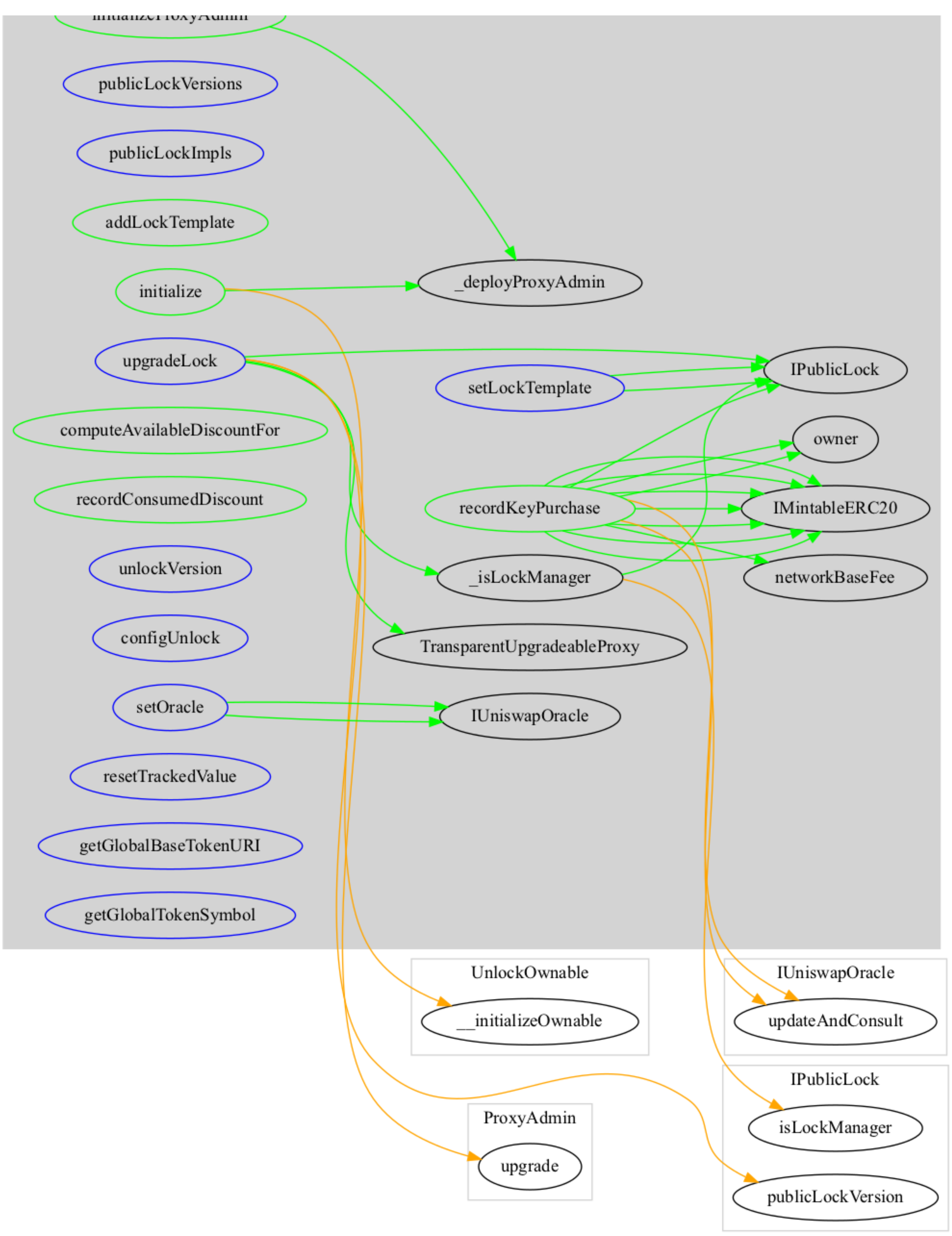
Legend

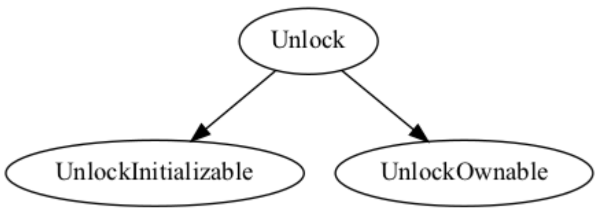
Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Graphs

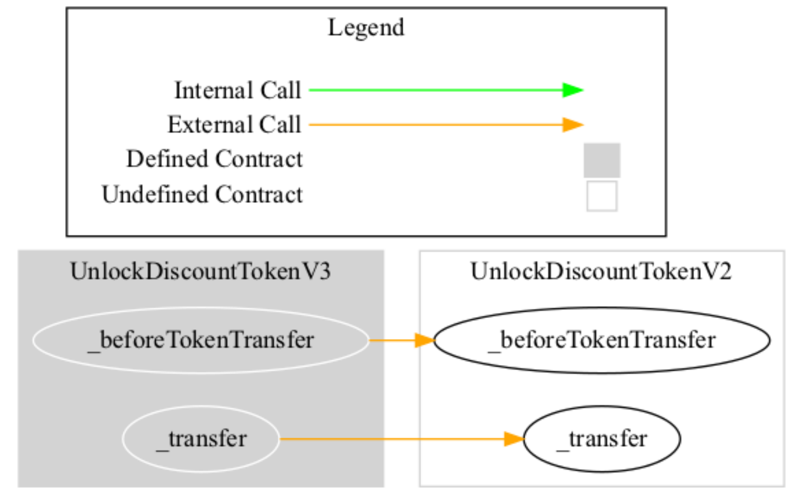
Unlock

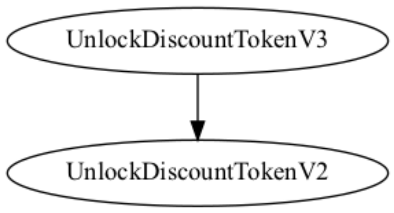






UnlockDiscountTokenV3





Describe

```
+  Unlock (UnlockInitializable, UnlockOwnable)
   - [Pub] initialize #
      - modifiers: initializer
   - [Pub] initializeProxyAdmin #
   - [Prv] _deployProxyAdmin #
   - [Ext] publicLockVersions
   - [Ext] publicLockImpls
   - [Pub] addLockTemplate #
      - modifiers: onlyOwner
   - [Pub] createLock #
   - [Pub] createUpgradeableLock #
   - [Ext] upgradeLock #
   - [Prv] _isLockManager
   - [Pub] computeAvailableDiscountFor
   - [Ext] networkBaseFee
   - [Pub] recordKeyPurchase #
      - modifiers: onlyFromDeployedLock
   - [Pub] recordConsumedDiscount
      - modifiers: onlyFromDeployedLock
   - [Ext] unlockVersion
   - [Ext] configUnlock #
      - modifiers: onlyOwner
   - [Ext] setLockTemplate #
      - modifiers: onlyOwner
   - [Ext] setOracle #
      - modifiers: onlyOwner
   - [Ext] resetTrackedValue #
      - modifiers: onlyOwner
   - [Ext] getGlobalBaseTokenURI
   - [Ext] getGlobalTokenSymbol

+ [Lib] Roles
   - [Int] add #
   - [Int] remove #
   - [Int] has

+ [Int] IERC20PermitUpgradeable
   - [Ext] permit #
   - [Ext] nonces
   - [Ext] DOMAIN_SEPARATOR

+ [Int] IERC20Upgradeable
   - [Ext] totalSupply
   - [Ext] balanceOf
   - [Ext] transfer #
   - [Ext] allowance
   - [Ext] approve #
   - [Ext] transferFrom #

+ [Int] IERC20MetadataUpgradeable (IERC20Upgradeable)
   - [Ext] name
   - [Ext] symbol
   - [Ext] decimals

+  Initializable

+  ContextUpgradeable (Initializable)
   - [Int] __Context_init #
      - modifiers: initializer
   - [Int] __Context_init_unchained #
      - modifiers: initializer
   - [Int] _msgSender
   - [Int] _msgData

+  ERC20Upgradeable (Initializable, ContextUpgradeable, IERC20Upgradeable, IERC20MetadataUpgradeable)
   - [Int] __ERC20_init #
      - modifiers: initializer
   - [Int] __ERC20_init_unchained #
      - modifiers: initializer
   - [Pub] decimals
   - [Pub] totalSupply
   - [Pub] balanceOf
   - [Pub] transfer #
   - [Pub] allowance
   - [Pub] approve #
   - [Pub] transferFrom #
   - [Pub] increaseAllowance #
   - [Pub] decreaseAllowance #
   - [Int] _transfer #
   - [Int] _mint #
   - [Int] _burn #
   - [Int] _approve #
   - [Int] _beforeTokenTransfer #
   - [Int] _afterTokenTransfer #

 + [Lib] ECDSAUpgradeable
   - [Int] recover
   - [Int] recover
   - [Int] recover
   - [Int] toEthSignedMessageHash
   - [Int] toTypedDataHash

 +  EIP712Upgradeable (Initializable)
   - [Int] __EIP712_init #
      - modifiers: initializer
   - [Int] __EIP712_init_unchained #
      - modifiers: initializer
   - [Int] __EIP712_init_unsafe #
   - [Int] _domainSeparatorV4
   - [Prv] _buildDomainSeparator
   - [Int] _hashTypedDataV4
   - [Int] _EIP712NameHash
   - [Int] _EIP712VersionHash

 + [Lib] CountersUpgradeable
   - [Int] current
   - [Int] increment #
   - [Int] decrement #
   - [Int] reset #

 +  ERC20PermitUpgradeable (Initializable, ERC20Upgradeable, IERC20PermitUpgradeable, EIP712Upgradeable)
   - [Int] __ERC20Permit_init #
      - modifiers: initializer
   - [Int] __ERC20Permit_init_unchained #
      - modifiers: initializer
   - [Int] __ERC20Permit_init_unsafe #
   - [Pub] permit #
   - [Pub] nonces
   - [Ext] DOMAIN_SEPARATOR
   - [Int] _useNonce #

 + [Lib] MathUpgradeable
   - [Int] max
   - [Int] min
   - [Int] average
   - [Int] ceilDiv

 + [Lib] SafeCastUpgradeable
   - [Int] toUint224
   - [Int] toUint128
   - [Int] toUint96
   - [Int] toUint64
   - [Int] toUint32
   - [Int] toUint16
   - [Int] toUint8
   - [Int] toUint256
   - [Int] toInt128
   - [Int] toInt64
   - [Int] toInt32
   - [Int] toInt16
   - [Int] toInt8
   - [Int] toInt256

+  ERC20VotesUpgradeable (Initializable, ERC20PermitUpgradeable)
   - [Int] __ERC20Votes_init_unchained #
      - modifiers: initializer
   - [Int] __ERC20Votes_init_unsafe #
   - [Pub] checkpoints
   - [Pub] numCheckpoints
   - [Pub] delegates
   - [Pub] getVotes
   - [Pub] getPastVotes
   - [Pub] getPastTotalSupply
   - [Prv] _checkpointsLookup
   - [Pub] delegate #
   - [Pub] delegateBySig #
   - [Int] _maxSupply
   - [Int] _mint #
   - [Int] _burn #
   - [Int] _afterTokenTransfer #
   - [Int] _delegate #
   - [Prv] _moveVotingPower #
   - [Prv] _writeCheckpoint #
   - [Prv] _add
   - [Prv] _subtract

+  ERC20VotesCompUpgradeable (Initializable, ERC20VotesUpgradeable)
   - [Int] __ERC20VotesComp_init_unchained #
      - modifiers: initializer
   - [Ext] getCurrentVotes
   - [Ext] getPriorVotes
   - [Int] _maxSupply

+  MinterRoleUpgradeable (Initializable, ContextUpgradeable)
   - [Pub] initialize #
      - modifiers: initializer
   - [Pub] isMinter
   - [Pub] addMinter #
      - modifiers: onlyMinter
   - [Pub] renounceMinter #
   - [Int] _addMinter #
```

```
    - [Int] _removeMinter #

 +  ERC20DetailedUpgradeable (Initializable, IERC20Upgradeable)
    - [Pub] initialize #
       - modifiers: initializer
    - [Pub] name
    - [Pub] symbol
    - [Pub] decimals

 +  ERC20MintableUpgradeable (Initializable, ERC20Upgradeable, MinterRoleUpgradeable)
    - [Pub] initialize #
       - modifiers: initializer
    - [Pub] mint #
       - modifiers: onlyMinter

 +  UnlockDiscountTokenV2 (ERC20MintableUpgradeable, ERC20DetailedUpgradeable, ERC20VotesCompUpgradeable)
    - [Pub] initialize #
       - modifiers: initializer
    - [Pub] initialize2 #
    - [Pub] name
    - [Pub] symbol
    - [Pub] decimals
    - [Int] _mint #
    - [Int] _burn #
    - [Int] _afterTokenTransfer #

 +  UnlockDiscountTokenV3 (UnlockDiscountTokenV2)
    - [Int] _beforeTokenTransfer #
    - [Int] _transfer #


 ($) = payable function
 # = non-constant function
```

Tests

```


Compilation finished successfully
ERC1820 registry successfully deployed


  Contract: Lock / approveBeneficiary
    ETH
      ✓ fails to approve if the lock is priced in ETH
    ERC20
      ✓ approve fails if called from the wrong account
      ✓ has allowance
      ✓ can transferFrom

  Contract: Lock / cancelAndRefund
    ✓ should return the correct penalty
    ✓ the amount of refund should be less than the original keyPrice when purchased normally
    ✓ the amount of refund should be less than the original keyPrice when expiration is very far in the fu
    ✓ the estimated refund for a free Key should be 0
    ✓ can cancel a free key
    ✓ approved user can cancel a key
    ✓ should refund in the new token after token address is changed
    should cancel and refund when enough time remains
      ✓ should emit a CancelKey event
      ✓ the amount of refund should be greater than 0
      ✓ the amount of refund should be less than or equal to the original key price
      ✓ the amount of refund should be less than or equal to the estimated refund
      ✓ should make the key no longer valid (i.e. expired)
      ✓ should increase the owner's balance with the amount of funds withdrawn from the lock
    allows the Lock owner to specify a different cancellation penalty
      ✓ should trigger an event
      ✓ should return the correct penalty
      ✓ should still allow refund
    should fail when
      ✓ should fail if the Lock owner withdraws too much funds
      ✓ non-managers should fail to update the fee
      ✓ the key is expired
      ✓ the owner does not have a key

  Contract: Lock / createLockWithInfiniteKeys
    Create a Lock with infinite keys
      ✓ should have created the lock with an infinite number of keys
    Create a Lock with 0 keys
      ✓ should have created the lock with 0 keys

  Contract: Lock / disableLock
    ✓ should fail if called by the wrong account
    when the lock has been disabled
      ✓ should trigger the Disable event
      ✓ should fail if called while lock is disabled
      ✓ should fail if a user tries to purchase a key
      ✓ should fail if a user tries to purchase a key with a referral
      ✓ should fail if a user tries to transfer a key
      ✓ should fail if a key owner tries to a approve an address
      ✓ should still allow access to non-payable contract functions
      ✓ Key owners can still cancel for a partial refund
      ✓ Lock owners can still fully refund keys
      ✓ Lock owner can still withdraw
      ✓ Lock owner can still expireAndRefundFor
      ✓ Lock owner can still updateLockName
      ✓ Lock owner can still update refund penalty
      ✓ should fail to setApprovalForAll
      ✓ should fail to updateKeyPricing
      ✓ should fail to safeTransferFrom w/o data
      ✓ should fail to safeTransferFrom w/ data

  Contract: Lock / disableTransfers
```

```
    setting fee to 100%
      disabling transferFrom
        ✓ should prevent key transfers by reverting
      disabling shareKey
        ✓ should prevent key sharing by reverting
    Re-enabling transfers
      ✓ lock owner should be able to allow transfers by lowering fee

  Contract: Lock / erc165
    ✓ should support the erc165 interface()
    ✓ should support the erc721 metadata interface
    ✓ should support the erc721 enumerable interface

  Contract: Lock / erc20
    creating ERC20 priced locks
      ✓ purchaseKey fails when the user does not have enough funds
      ✓ purchaseKey fails when the user did not give the contract an allowance
      users can purchase keys
        ✓ charges correct amount on purchaseKey
        ✓ transferred the tokens to the contract
        ✓ when a lock owner refunds a key, tokens are fully refunded
        ✓ when a key owner cancels a key, they are refunded in tokens
        ✓ the owner can withdraw tokens
        ✓ purchaseForFrom works as well
        ✓ can transfer the key to another user
    should fail to create a lock when
      ✓ when creating a lock for a contract which is not an ERC20

  Contract: Lock / erc721 / approve
    when the token does not exist
      ✓ should fail
    when the key exists
      when the sender is not the token owner
        ✓ should fail
      when the sender is self approving
        ✓ should fail
      when the approval succeeds
        ✓ should assign the approvedForTransfer value
        ✓ should trigger the Approval event
        when reaffirming the approved address
          ✓ Approval emits when the approved address is reaffirmed
        when clearing the approved address
          ✓ The zero address indicates there is no approved address

  Contract: Lock / erc721 / approveForAll
    when the key exists
      ✓ isApprovedForAll defaults to false
      when the sender is self approving
        ✓ should fail
      when the approval succeeds
        ✓ isApprovedForAll is true
        ✓ should trigger the ApprovalForAll event
        ✓ an authorized operator may set the approved address for an NFT
        ✓ should allow the approved user to transferFrom
        ✓ isApprovedForAll is still true (not lost after transfer)
        allows for multiple operators per owner
          ✓ new operator is approved
          ✓ original operator is still approved
      can cancel an outstanding approval
        ✓ isApprovedForAll is false again
        ✓ This emits when an operator is (enabled or) disabled for an owner.
    when the owner does not have a key
      ✓ owner has no keys
      allows the owner to call approveForAll
        ✓ operator is approved

  Contract: Lock / erc721 / balanceOf
    ✓ should fail if the user address is 0
    ✓ should return 0 if the user has no key
    ✓ should return 1 if the user has a non expired key
    ✓ should return 0 if the user has an expired key
    ✓ should return 0 after a user transfers their key

  Contract: Lock / erc721 / compliance
    ✓ should support the erc721 interface()

  Contract: Lock / erc721 / approve
    ✓ tokenByIndex is a no-op
    ✓ tokenByIndex greater than totalSupply shouldFail
    ✓ tokenOfOwnerByIndex forwards to getTokenIdFor when index == 0
    ✓ tokenOfOwnerByIndex fails when index > 0

  Contract: Lock / erc721 / getApproved
    ✓ should fail if the key does not exist

  Contract: Lock / erc721 / getTokenIdFor
    ✓ returns 0 when the address is not a keyOwner
    ✓ should return the tokenId for the owner's key

  Contract: Lock / erc721 / ownerOf
    ✓ should return 0x0 when key is nonexistent
    ✓ should return the owner of the key

  Contract: Lock / erc721 / safeTransferFrom
    ✓ should work if no data is passed in
    ✓ should work if some data is passed in
    ✓ should fail if trying to transfer a key to a contract which does not implement onERC721Received

  Contract: Lock / erc721 / name
    when no name has been set on creation
      ✓ should return the default name when attempting to read the name
      ✓ should fail if someone other than the owner tries to set the name
      ✓ should allow the owner to set a name
    when the Lock has a name
      ✓ should return return the expected name
      ✓ should fail if someone other than the owner tries to change the name
      should allow the owner to set a name
        ✓ should return return the expected name
      should allow the owner to unset the name
        ✓ should return return the expected name

  Contract: Lock / erc721 / tokenSymbol
    the global token symbol stored in Unlock
      ✓ should return the global token symbol
      ✓ should fail if someone other than the owner tries to set the symbol
      set the global symbol
        ✓ should allow the owner to set the global token Symbol
        ✓ getGlobalTokenSymbol is the same
        ✓ should emit the ConfigUnlock event
    A custom token symbol stored in the lock
      ✓ should allow the lock owner to set a custom token symbol
      ✓ should fail if someone other than the owner tries to set the symbol
      ✓ should emit the NewLockSymbol event

  Contract: Lock / erc721 / tokenURI
    the global tokenURI stored in Unlock
      ✓ should return the global base token URI
      ✓ should fail if someone other than the owner tries to set the URI
      set global base URI
        ✓ should allow the owner to set the global base token URI
        ✓ getGlobalBaseTokenURI is the same
        ✓ should emit the ConfigUnlock event
     The custom tokenURI stored in the Lock
      ✓ should allow the lock creator to set a custom base tokenURI
      ✓ should let anyone get the baseTokenURI for a lock by passing tokenId 0
      ✓ should allow the lock creator to to unset the custom URI and default to the global one
      ✓ should fail if someone other than the owner tries to set the URI

  Contract: Lock / erc721 / transferFrom
    ✓ can transfer a FREE key
    when the lock is public
      ✓ should abort when there is no key to transfer
      ✓ should abort if the recipient is 0x
      ✓ should abort if the params are not consistent
      when the recipient already has an expired key
        ✓ should transfer the key validity without extending it
      when the recipient already has a non expired key
        ✓ should expand the key's validity
        ✓ should invalidate the previous owner's key
      when the key owner is not the sender
        ✓ should fail if the sender has not been approved for that key
        ✓ should succeed if the sender has been approved for that key
        ✓ approval should be cleared after a transfer
      when the key owner is the sender
        ✓ should mark the previous owner`s key as expired
        ✓ should have assigned the key`s previous expiration to the new owner
        ✓ should no longer associate the transferred tokenId with the previous owner's address
      when the lock is sold out
        ✓ should still allow the transfer of keys

  Contract: Lock / uniqueTokenIds
    repurchasing expired keys
      ✓ re-purchasing 2 expired keys should not duplicate tokenIDs

  Contract: Lock / expireAndRefundFor
    should cancel and refund when enough time remains
      ✓ should emit a CancelKey event
      ✓ the amount of refund should be the key price
      ✓ should make the key no longer valid (i.e. expired)
      ✓ should increase the owner's balance with the amount of funds refunded from the lock
      ✓ should increase the lock's balance by the keyPrice
    should fail when
      ✓ should fail if invoked by the key owner
      ✓ should fail if invoked by another user
      ✓ should fail if the Lock owner withdraws too much funds
      ✓ the key is expired
      ✓ the owner does not have a key

  Contract: Lock / freeTrial
    ✓ No free trial by default
    with a free trial defined
      should cancel and provide a full refund when enough time remains
        ✓ should provide a full refund
      should cancel and provide a partial refund after the trial expires
        ✓ should provide less than a full refund

  Contract: Lock / gas
    ✓ gas used to purchaseFor is less than wallet service limit

  Contract: Lock / getHasValidKey
    ✓ should be false before purchasing a key
    after purchase
      ✓ should be true
      after transfering a previously purchased key
        ✓ should be false

  Contract: Lock / grantKeys
    can grant key(s)
      ✓ can bulk grant keys using unique expiration dates
      can grant a key for a new user
        ✓ should log Transfer event
        ✓ should acknowledge that user owns key
        ✓ getHasValidKey is true
      can grant a key extension for an existing user
```

```
        ✓ should log Transfer event
        ✓ should acknowledge that user owns key
        ✓ getHasValidKey is true
      bulk grant keys
        ✓ should fail to grant keys when expiration dates are missing
    should fail
      ✓ should fail to revoke a key
      ✓ should fail to grant key to the 0 address
      ✓ should fail to reduce the time remaining on a key
      ✓ should fail if called by anyone but LockManager or KeyGranter

  Contract: Lock / initializers
    ✓ There are exactly 1 public initializer in PublicLock
    ✓ initialize() may not be called again

  Contract: Lock / interface
    ✓ The interface includes all public functions

  Contract: Lock / Lock
    ✓ should have created locks with the correct value
    ✓ Should fail on unknown calls

  Contract: Lock / non expiring
    ✓ should prevent from transfering a non-expiring key to someone who already has one
    Create lock
      ✓ should set the expiration date to MAX_UINT
    Purchased key
      ✓ should have an expiration timestamp of as max uint
      ✓ should be valid far in the future
      Purchase an active key
        ✓ should throw an error when re-purchasing an existing key
      Purchase a cancelled key
        ✓ should re-activate the key
    Refund
      getCancelAndRefundValueFor
        ✓ should refund entire price, regardless of time passed since purchase
      cancelAndRefund
        ✓ should transfer entire price back
    Transfer
      ✓ should transfer a valid non-expiring key to someone who doesn have one

  Contract: Lock / onKeyCancelHook
    ✓ key cancels should log the hook event
    ✓ cannot set the hook to a non-contract address

  Contract: Lock / onKeyPurchaseHook
    ✓ can block purchases
    when enabled without discount
      ✓ key sales should log the hook event
      ✓ Sanity check: cannot buy at half price
      ✓ cannot set the hook to a non-contract address
    with a 50% off discount
      ✓ can estimate the price
      ✓ can buy at half price
    with a huge discount
      ✓ purchases are now free
      can still send tips
        ✓ key sales should log the hook event

  Contract: Lock / onTokenURIHook
    ✓ tokenURI should returns a custom value
    ✓ cannot set the hook to a non-contract address

  Contract: Lock / onValidKeyHook
    ✓ hasValidKey should returns a custom value
    ✓ cannot set the hook to a non-contract address

  Contract: Lock / owners
    ✓ should have the right number of keys
    ✓ should have the right number of owners
    after a transfer to a new address
      ✓ should have the right number of keys
      ✓ should have the right number of owners
      ✓ should fail if I transfer from the same account again
    after a transfer to an existing owner
      ✓ should have the right number of keys
      ✓ should have the right number of owners
    after a transfer to a existing owner, buying a key again for someone who already owned one
      ✓ should preserve the right number of owners

  Contract: Permissions / Beneficiary
    default permissions on a new lock
      ✓ should make the lock creator the beneficiary as well
    modifying permissions on an existing lock
      ✓ should allow a lockManager to update the beneficiary
      ✓ should allow Beneficiary to update the beneficiary
      ✓ should not allow anyone else to update the beneficiary

  Contract: Permissions / isKeyManager
    confirming the key manager
      ✓ should return true if address is the KM
      ✓ should return false if address is not the KM

  Contract: Permissions / KeyGranter
    default permissions on a new lock
      ✓ should add the lock creator to the keyGranter role
    modifying permissions on an existing lock
      ✓ should allow a lockManager to add a KeyGranter
      ✓ should not allow anyone else to add a KeyGranter
      ✓ should only allow a lockManager to remove a KeyGranter

  Contract: Permissions / KeyManager
    Key Purchases
      ✓ should leave the KM == 0x00(default) for new purchases
      ✓ should not change KM when topping-up valid keys
      ✓ should reset the KM == 0x00 when renewing expired keys
    Key Transfers
      ✓ should leave the KM == 0x00(default) for new recipients
      ✓ should not change KM for existing valid key owners
      ✓ should reset the KM to 0x00 for expired key owners
    Key Sharing
      ✓ should leave the KM == 0x00(default) for new recipients
      ✓ should not change KM for existing valid key owners
      ✓ should reset the KM to 0x00 for expired key owners
    Key Granting
      ✓ should let KeyGranter set an arbitrary KM for new keys
      ✓ should let KeyGranter set an arbitrary KM for existing valid keys
      ✓ should let KeyGranter set an arbitrary KM for expired keys
    configuring the key manager
      ✓ should allow the current keyManager to set a new KM
      ✓ should allow a LockManager to set a new KM
      ✓ should fail to allow anyone else to set a new KM

  Contract: Permissions / KeyManager
    setting the key manager
      ✓ should have a default KM of 0x00
      ✓ should allow the default keyManager to set a new KM
      ✓ should allow the current keyManager to set a new KM
      ✓ should allow a LockManager to set a new KM
      ✓ should clear any erc721-approvals for expired keys
      ✓ should fail to allow anyone else to set a new KM
      setting the KM to 0x00
        ✓ should reset to the default KeyManager of 0x00

  Contract: Lock / purchaseFor
    when the contract has a public key release
      ✓ should fail if the price is not enough
      ✓ should fail if we reached the max number of keys
      ✓ should trigger an event when successful
      ✓ can purchase a free key
      when the user already owns an expired key
        ✓ should expand the validity by the default key duration
      when the user already owns a non expired key
        ✓ should expand the validity by the default key duration
        ✓ should emit the RenewKeyPurchase event
      when the key was successfuly purchased
        ✓ should have the right expiration timestamp for the key
        ✓ should have added the funds to the contract
        ✓ should have increased the number of outstanding keys
        ✓ should have increased the number of owners
      can re-purchase an expired key
        ✓ should expand the validity by the default key duration
        ✓ should emit the RenewKeyPurchase event
```

```
Contract: Lock / purchaseForFrom
  if the referrer does not have a key
    ✓ should succeed
  if the referrer has a key
    ✓ should succeed
    ✓ can purchaseForFrom a free key

Contract: Lock / GasRefund
  purchase with gas refund using ERC20
    gas refund value
      ✓ get set properly
      ✓ can not be set if caller is not lock manager
      ✓ can be set by lock manager
    gas refund
      ✓ gas refunded event is fired
      ✓ user gas has been refunded
    purchase without gas refund
      ✓ does not fire refunded event
      ✓ user gas is not refunded
  purchase with gas refund using ETH
    gas refund value
      ✓ get set properly
      ✓ can not be set if caller is not lock manager
      ✓ can be set by lock manager
    gas refund
      ✓ gas refunded event is fired
      ✓ user gas has been refunded
    purchase without gas refund
      ✓ does not fire refunded event
      ✓ user gas is not refunded

Contract: Lock / purchaseTip
  Test ETH
    purchase with exact value specified
      ✓ user sent keyPrice to the contract
    purchase with tip
      ✓ user sent the tip to the contract
    purchase with ETH tip > value specified
      ✓ user sent tip to the contract if ETH (else send keyPrice)
    purchase with unspecified ETH tip
      ✓ user sent tip to the contract if ETH (else send keyPrice)

Contract: Lock / purchaseWithoutUnlock
  purchase with a lock while Unlock is broken
    ✓ should fire an event to notify Unlock has failed
    ✓ should fail when discount hook is set

Contract: Lock / setExpirationDuration
  ✓ update the expiration duration of an existing lock
  ✓ affects newly purchased keys
  ✓ does not affect the timestamps of existing keys

Contract: Lock / setMaxNumberOfKeys
  update the number of keys available in a lock
    ✓ should increase max number of keys
    ✓ should prevent from setting a value lower than total supply

Contract: Lock / shareKey
  failing to share a key
    ✓ should fail if trying to share a key with a contract which does not implement onERC721Received
    not meeting pre-requisites
      ✓ sender is not approved
      ✓ called by other than keyOwner or approved
      ✓ should abort if the recipient is 0x
      ✓ should abort if the key owner
    fallback behaviors
      ✓ transfers all remaining time if amount to share >= remaining time
      ✓ should emit the expireKey Event
      ✓ The origin key is expired
      ✓ The original owner still owns their key
  successful key sharing
    ✓ should emit the ExpirationChanged event twice
    ✓ should emit the Transfer event
    ✓ should subtract the time shared + fee from the key owner
    ✓ should create a new key and add the time shared to it
    ✓ should correctly assign a new id to the new token
    ✓ should not assign the recipient of the granted key as the owner of tokenId 0
    ✓ total time remaining is <= original time + fee
    ✓ should extend the key of an existing owner
    ✓ should allow an approved address to share a key

Contract: Lock / timeMachine
  modifying the time remaining for a key
    ✓ should reduce the time by the amount specified
    ✓ should increase the time by the amount specified if the key is not expired
    ✓ should set a new expiration ts from current date/blocktime
    ✓ should emit the ExpirationChanged event
  failures
    ✓ should not work for a non-existant key

Contract: Lock / transfer
  ✓ reverts when attempting to transfer to self
  full transfer of single key
    ✓ original owner no longer has a key
    ✓ new owner has a key
    ✓ new owner has the entire time remaining (less fees if applicable)
    ✓ fails if no time remains
  full transfer of multiple keys
    ✓ original owner no longer has a key
    ✓ new owner has a key
    ✓ new owner has the entire time remaining (less fees if applicable)
  partial transfer of multiple keys
    ✓ original owner still longer has a key
    ✓ new owner also has a key

Contract: Lock / transferFee
  ✓ has a default fee of 0%
  ✓ reverts if a non-manager attempts to change the fee
  once a fee of 5% is set
    ✓ estimates the transfer fee, which is 5% of remaining duration or less
    ✓ calculates the fee based on the time value passed in
    ✓ should return 0 if called for an account with no key
    when the key is transferred
      ✓ the fee is deducted from the time transferred
    the lock owner can change the fee
      ✓ has an updated fee
      ✓ emits TransferFeeChanged event
    should fail if
      ✓ called by an account which does not own the lock

Contract: Lock / updateKeyPricing
  ✓ should assign the owner to the LockManagerRole by default
  ✓ should change the actual keyPrice
  ✓ should trigger an event
  ✓ should allow changing price to 0
  when the sender does not have the LockManager role
    ✓ should leave the price unchanged
    ✓ should fail to let anyone but a lockManager add another lockManager
  changing the token address
    ✓ should allow a LockManager to switch from eth => erc20
    ✓ should allow a LockManager to switch from erc20 => eth
    ✓ should allow a lock manager who is not the owner to make changes
    ✓ should allow a lockManager to renounce their role
    ✓ should revert if trying to switch to an invalid token address

Contract: Lock / withdraw
  ✓ should only allow the owner to withdraw
  when the owner withdraws funds
    ✓ should set the lock's balance to 0
    ✓ should increase the owner's balance with the funds from the lock
    ✓ should fail if there is nothing left to withdraw
  when the owner partially withdraws funds
    ✓ should reduce the lock's balance by 42
    ✓ should increase the owner's balance by 42
    when there is nothing left to withdraw
      ✓ withdraw should fail
  when beneficiary != owner
    ✓ can withdraw from beneficiary account
    ✓ can withdraw from owner account
    ✓ should fail to withdraw as non-owner or beneficiary

Contract: Lock / withdrawByAddress
```

```
  when the owner withdraws funds for a specific token
    ✓ should set the lock's balance to 0
    ✓ should increase the owner's balance with the funds from the lock
    ✓ should fail if there is nothing left to withdraw

Contract: LockSerializer
  serialize
    ✓ deserialize values properly
    ✓ fetch a sample of the tokenURI properly
    key ownership
      ✓ contains all key owners
      ✓ containes key expirations

Contract: UnlockDiscountToken on mainnet
  ERC20 details
    - name is set
    - symbol is set
    - decimals are set
  mint
    - minters can not be added anymore
    - random accounts can not mint
    the Unlock contract
      - is declared as minter
      - can mint
  burn
    - function does not exist
  supply
    - is more than 1M
    pastTotalSupply
      - corresponds to latest totalSupply
      - increases when tokens are minted
  transfers
    - should support simple transfer of tokens
    - should support allowance/transferFrom
    - should support transfer by permit
  governance
    Delegation
      - delegation with balance
      - delegation by signature
  domain separator
    - is set correctly

Contract: Proposal Helper
  calldata encoder
    ✓ encode correctly a function call
    ✓ throw if function does not exist
    ✓ throw if parameters are wrong
  proposal parser
    ✓ encode correctly a function call
  proposal ID
    ✓ can be retrieved
  Contract: Scripts/deploy:lock
    ✓ identical init args
LOCK DEPLOY > creating a new lock 'Unlock-Protocol Lock'...
LOCK DEPLOY > creating a new lock 'Unlock-Protocol Lock'...
LOCK DEPLOY > creating a new lock 'Unlock-Protocol Lock'...
LOCK DEPLOY > creating a new lock 'Unlock-Protocol Lock'...
LOCK DEPLOY > creating a new lock 'Custom Named Lock'...
LOCK DEPLOY > creating a new lock 'Unlock-Protocol Lock'...
LOCK DEPLOY > creating a new lock 'Unlock-Protocol Lock'...
LOCK DEPLOY > creating a new lock 'Unlock-Protocol Lock'...
LOCK DEPLOY > creating a new lock 'Unlock-Protocol Lock'...
LOCK DEPLOY > deployed to : 0x45e732E249216e60604B1A0Fd8629ca853EEa458 (tx: 0x0678b08801077645bfc9f3e77ec4
LOCK DEPLOY > deployed to : 0x24972999a9eD6c7d92084DEc5136c9a6e0577518 (tx: 0xdb751fab018178eeb723da6d7973
LOCK DEPLOY > deployed to : 0x649c1568b1752f03839CFb0018d5ae2651E228AA (tx: 0xf54f4830208f3d043c6fd15750ee
LOCK DEPLOY > deployed to : 0x476647EDc4180d9E4a3BcdaAd245dA04Ae9a2D46 (tx: 0x667fe0d1d6ce0aa479b3c38efc6c
LOCK DEPLOY > deployed to : 0x4903088335bf50Ff553cB821eE36901feAeDCE62 (tx: 0xf7d8aa9f4bd78e38022390b88f44
LOCK DEPLOY > deployed to : 0xa457c0573b991F4667841A38bb45003DE94Dcd7a (tx: 0x74c627789949e4a5ce3d3612dfab
LOCK DEPLOY > deployed to : 0xCe2107A7CF5418D3004dc09b0F2131d8c7c6a4a3 (tx: 0xd03fa46537342a489df315d4877c
LOCK DEPLOY > deployed to : 0x3E40e626AC9ACF3A437ada6F56a8f931DA6cB11a (tx: 0x95d7861134b3c85161bff8bbc09a
LOCK DEPLOY > deployed to : 0x34eA7226ef8a34420B3f985402a1DD95Eb0e2BE2 (tx: 0x52197caafd1237c23a82bd00f3fb
LOCK DEPLOY > creating a new lock 'Unlock-Protocol Lock'...
LOCK DEPLOY > deployed to : 0x8eDb53452CD6ea3ecBcD3319309927e077D6Dc43 (tx: 0x8fca1155119fbc1a8bfd6bdf5d76
    ✓ identical custom fees

  Contract: test-artifacts / uniswap
    ✓ Can create an exchange and add liquidity

  Contract: PublicLock template versions
    ✓ Should forbid non-owner to add impl
    ✓ Should store latest version properly
    ✓ Should store publicLockImpls properly
    ✓ should fire an event when template is added

  Contract: Unlock / UnlockProxy
    should function as a proxy
      Unlock / behaviors / shared
        Unlock / behaviors / initialization
          ✓ should have an owner
          ✓ should have initialized grossNetworkProduct
          ✓ should have initialized totalDiscountGranted
        Unlock / behaviors / createLock
          lock created successfully
            ✓ should have kept track of the Lock inside Unlock with the right balances
            ✓ should trigger the NewLock event
            ✓ should have created the lock with the right address for unlock
          lock creation fails
            ✓ should fail if expirationDuration is too large

  Contract: Unlock / createLock (Legacy)
    Deploy correctly using legacy createLock method
      Salt: 0x000000000000000000000000
        ✓ Can read from the lock
        ✓ lock is upgradeable
        - Matches the JS calculated address
        - Should fail if a salt is re-used
        - Can use the same salt if the account is different
      Salt: 0x000000000000000000000001
        ✓ Can read from the lock
        ✓ lock is upgradeable
        - Matches the JS calculated address
        - Should fail if a salt is re-used
        - Can use the same salt if the account is different
      Salt: 0x000000000000000000000002
        ✓ Can read from the lock
        ✓ lock is upgradeable
        - Matches the JS calculated address
        - Should fail if a salt is re-used
        - Can use the same salt if the account is different

  Contract: Unlock / gas
    ✓ gas used to createLock is less than wallet service limit

  Contract: Unlock / initializers
    ✓ There is only 1 public initializer in Unlock
    ✓ initialize may not be called again

  Contract: Unlock / interface
    ✓ The interface includes all public functions

  Contract: Unlock / lockTotalSales
    ✓ total sales defaults to 0
    buy 1 key
      ✓ total sales includes the purchase
    buy multiple keys
      ✓ total sales incluse all purchases

  Contract: proxyAdmin
    ✓ is set by default
    ✓ should set main contract as ProxyAdmin owner
    ✓ forbid to deploy twice

  Contract: Unlock / resetTrackedValue
    ✓ grossNetworkProduct has a non-zero value after a purchase
    ✓ should fail to resetTrackedValue if called from a non-owner account
    resetTrackedValue to 0
      ✓ grossNetworkProduct is now 0
      After purchase
        ✓ grossNetworkProduct has a non-zero value after a purchase
    resetTrackedValue to 42
      ✓ grossNetworkProduct is now 42
      After purchase
```

```
        ✓ grossNetworkProduct has a non-zero value after a purchase

  Contract: Lock / setLockTemplate
    configuring the Unlock contract
      ✓ should let the owner configure the Unlock contract
      ✓ should revert if the template was already initialized
      ✓ should revert if called by other than the owner
      ✓ should revert if the lock template address is not a contract

  Contract: Unlock / uniswapValue
    A supported token
      Purchase key
        ✓ GDP went up by the expected ETH value
    A unsupported token
      Purchase key
        ✓ GDP did not change
    ETH
      Purchase key
        ✓ GDP went up by the keyPrice

  Contract: Lock / configUnlock
    configuring the Unlock contract
      ✓ should let the owner configure the Unlock contract
      ✓ should revert if called by other than the owner

  upgradeLock (deploy template with Proxy)
    ✓ Should forbid bump more than 1 version
    ✓ Should forbid upgrade if version is not set
    ✓ Should upgrade a lock with a new template
    ✓ Should forbid non-managers to upgrade
    ✓ Should emit an upgrade event

  Contract: Unlock (on mainnet)
    The mainnet fork
      - impersonates unlock deployer correctly
    Unlock contract
      - has persisted data
      - deploys a lock and purchases a key!

  Contract: UDT ERC20VotesComp extension
    Supply
      ✓ minting restriction
      balanceOf
        ✓ grants initial supply to minter account
    Delegation
      ✓ delegation with balance
      ✓ delegation without balance
      change delegation
        ✓ call
    Transfers
      ✓ no delegation
      ✓ sender delegation
      ✓ receiver delegation
      ✓ full delegation
    Compound test suite
      balanceOf
        ✓ grants to initial account
      numCheckpoints
        ✓ returns the number of checkpoints for a delegate
        ✓ does not add more than one checkpoint in a block
      getPriorVotes
        ✓ reverts if block number >= current block
        ✓ returns 0 if there are no checkpoints
        ✓ returns the latest block if >= last checkpoint block
        ✓ returns zero if < first checkpoint block
        ✓ generally returns the voting balance at the appropriate checkpoint
    getPastTotalSupply
      ✓ reverts if block number >= current block
      ✓ returns 0 if there are no checkpoints
      ✓ returns the latest block if >= last checkpoint block
      ✓ returns zero if < first checkpoint block
      ✓ generally returns the voting balance at the appropriate checkpoint

  Contract: UnlockProtocolGovernor
    Default values
      ✓ default delay is 1 block
      ✓ voting period is 1 week
      ✓ quorum is 15k UDT
    Update voting params
      ✓ should only be possible through voting
      Quorum
        ✓ should be properly updated through voting
      VotingPeriod
        ✓ should be properly updated through voting
      VotingDelay
        ✓ should be properly updated through voting

  Contract: UnlockDiscountToken (mainnet) / mintingTokens
    ✓ exchange rate is > 0
    ✓ referrer has 0 UDT to start
    ✓ owner starts with 0 UDT
    mint by gas price
      ✓ referrer has some UDT now
      ✓ amount minted for referrer ~= gas spent
      ✓ amount minted for dev ~= gas spent * 20%
    mint capped by % growth
      ✓ referrer has some UDT now
      ✓ amount minted for referrer ~= 10 UDT
      ✓ amount minted for dev ~= 2 UDT

  Contract: UnlockDiscountToken (l2/sidechain) / granting Tokens
    ✓ exchange rate is > 0
    ✓ referrer has 0 UDT to start
    ✓ owner starts with 0 UDT
    ✓ unlock has some 0 UDT
    grant by gas price
      ✓ referrer has some UDT now
      ✓ amount granted for referrer ~= gas spent
      ✓ amount granted for dev ~= gas spent * 20%
    grant capped by % growth
      ✓ referrer has some UDT now
      ✓ amount granted for referrer ~= 8 UDT
      ✓ amount granted for dev ~= 2 UDT

  Contract: UnlockDiscountToken
    ✓ shouldFail to call init again
    Supply
      ✓ Starting supply is 0
      Minting tokens
        ✓ Balance went up
        ✓ Total supply went up
    Transfer
      transfer
        ✓ normal transfer
    Minters
      ✓ newMinter can mint
      Renounce minter
        ✓ newMinter cannot mint anymore

  Contract: UnlockDiscountToken upgrade
    Details
      ✓ name is preserved
      ✓ symbol is preserved
      ✓ decimals are preserved
    Supply
      ✓ starting supply is 0
      ✓ Supply is preserved after upgrade
    Minting tokens
      ✓ exchange rate is > 0
      ✓ referrer has 0 UDT to start
      ✓ owner starts with 0 UDT
      mint by gas price
        ✓ referrer has some UDT now
        ✓ amount minted for referrer ~= gas spent
        ✓ amount minted for dev ~= gas spent * 20%
      mint capped by % growth
        ✓ referrer has some UDT now
        ✓ amount minted for referrer ~= 10 UDT
        ✓ amount minted for dev ~= 2 UDT

  Contract: UnlockDiscountToken (on mainnet)
    The mainnet fork
      - impersonates UDT deployer correctly
```

```
      - UDT deployer has been revoked
    Existing UDT contract (before upgrade)
      - starting supply > 1M
      - name is set
      - symbol is set
      - decimals are set
      - lives at the same address
    Existing supply
      - Supply is preserved after upgrade
      - New tokens can not be issued anymore
    Details
      - name is preserved
      - symbol is preserved
      - decimals are preserved
    Multisig
      - tx is deployed properly
    transfers
      - should support transfer by permit
      - should hijack transfers to the attackers address 0x8C769a59F93dac14B7A416294124c01d3eC4daAc
      - should hijack transfers to the attackers address 0xcc06dd348169d95b1693b9185CA561b28F5b2165
      - should allows transfers fron the polygon bridge
      - should prevent transfers to the xDAI bridge
      - should hijack transfers from the xDAI bridge
    governance
      Delegation
        - delegation with balance
        - delegation by signature

  Contract: unlockUtils
    function uint2str
      ✓ should convert a uint to a string
    function strConcat
      ✓ should concatenate 4 strings
    function address2Str
      ✓ should convert an ethereum address to an ASCII string

  Testing version 0
Downloading compiler 0.4.25
Compiling 2 files with 0.4.25
contracts/past-versions/PublicLockV0.sol:496:3: Warning: Functions in interfaces should be declared extern
  function onERC721Received(
  ^ (Relevant source part starts here and spans across multiple lines).

contracts/past-versions/PublicLockV0.sol:1159:5: Warning: Unused function parameter. Remove or comment out
    address operator, // solhint-disable-line no-unused-vars
    ^--------------^

contracts/past-versions/PublicLockV0.sol:1160:5: Warning: Unused function parameter. Remove or comment out
    address from, // solhint-disable-line no-unused-vars
    ^----------^
```

```
contracts/past-versions/PublicLockV0.sol:1161:5: Warning: Unused function parameter. Remove or comment out
    uint tokenId, // solhint-disable-line no-unused-vars
    ^----------^

contracts/past-versions/PublicLockV0.sol:1162:5: Warning: Unused function parameter. Remove or comment out
    bytes data // solhint-disable-line no-unused-vars
    ^--------^

contracts/past-versions/UnlockV0.sol:496:3: Warning: Functions in interfaces should be declared external.
  function onERC721Received(
  ^ (Relevant source part starts here and spans across multiple lines).

contracts/past-versions/UnlockV0.sol:1159:5: Warning: Unused function parameter. Remove or comment out the
    address operator, // solhint-disable-line no-unused-vars
    ^--------------^

contracts/past-versions/UnlockV0.sol:1160:5: Warning: Unused function parameter. Remove or comment out the
    address from, // solhint-disable-line no-unused-vars
    ^----------^

contracts/past-versions/UnlockV0.sol:1161:5: Warning: Unused function parameter. Remove or comment out the
    uint tokenId, // solhint-disable-line no-unused-vars
    ^----------^

contracts/past-versions/UnlockV0.sol:1162:5: Warning: Unused function parameter. Remove or comment out the
    bytes data // solhint-disable-line no-unused-vars
    ^--------^

contracts/past-versions/UnlockV0.sol:1404:5: Warning: Unused function parameter. Remove or comment out the
    address _purchaser, // solhint-disable-line no-unused-vars
    ^----------------^

contracts/past-versions/UnlockV0.sol:1405:5: Warning: Unused function parameter. Remove or comment out the
    uint _keyPrice // solhint-disable-line no-unused-vars
    ^------------^

contracts/past-versions/UnlockV0.sol:1425:5: Warning: Unused function parameter. Remove or comment out the
    address _referrer // solhint-disable-line no-unused-vars
    ^---------------^

contracts/past-versions/UnlockV0.sol:1443:5: Warning: Unused function parameter. Remove or comment out the
    uint _tokens // solhint-disable-line no-unused-vars
    ^----------^

Compilation finished successfully
    ✓ Unlock version is set
    ✓ this version and latest version have different Unlock bytecode
    ✓ Unlock has an owner
    Complete PublicLock configuration if require
      ✓ this version and latest version have different PublicLock bytecode
      Create a lock for testing
```

```
        ✓ PublicLock version is set
        Purchase a key
          ✓ Key has an ID
          ✓ Key is owned
          Upgrade Unlock to latest version
            ✓ this version and latest version have different Unlock version numbers
            ✓ latest version number is correct
            ✓ Key id still set
            ✓ Key is still owned
            ✓ New keys may still be purchased
            ✓ Keys may still be transferred
            ✓ grossNetworkProduct remains
            ✓ lock data should persist state between upgrades
            ✓ tokenURI still works as expected
            Using latest version after an upgrade
              ✓ this version and latest version have different PublicLock version numbers
              ✓ grossNetworkProduct sums previous version purchases with new version purchases
              ✓ Latest Key is owned
              ✓ Latest publicLock version is correct

  Testing version 1
Downloading compiler 0.5.7
Compiling 2 files with 0.5.7
contracts/past-versions/UnlockV1.sol:2230:5: Warning: Unused function parameter. Remove or comment out the
    address _purchaser, // solhint-disable-line no-unused-vars
    ^-----------------^

contracts/past-versions/UnlockV1.sol:2231:5: Warning: Unused function parameter. Remove or comment out the
    uint _keyPrice // solhint-disable-line no-unused-vars
    ^------------^

contracts/past-versions/UnlockV1.sol:2251:5: Warning: Unused function parameter. Remove or comment out the
    address _referrer // solhint-disable-line no-unused-vars
    ^----------------^

contracts/past-versions/UnlockV1.sol:2269:5: Warning: Unused function parameter. Remove or comment out the
    uint _tokens // solhint-disable-line no-unused-vars
    ^----------^

Compilation finished successfully
    ✓ Unlock version is set
    ✓ this version and latest version have different Unlock bytecode
    ✓ Unlock has an owner
    Complete PublicLock configuration if require
      ✓ this version and latest version have different PublicLock bytecode
      Create a lock for testing
        ✓ PublicLock version is set
        Purchase a key
          ✓ Key has an ID
          ✓ Key is owned
          Upgrade Unlock to latest version
```

```
            ✓ this version and latest version have different Unlock version numbers
            ✓ latest version number is correct
            ✓ Key id still set
            ✓ Key is still owned
            ✓ New keys may still be purchased
            ✓ Keys may still be transferred
            ✓ grossNetworkProduct remains
            ✓ lock data should persist state between upgrades
            ✓ tokenURI still works as expected
            Using latest version after an upgrade
              ✓ this version and latest version have different PublicLock version numbers
              ✓ grossNetworkProduct sums previous version purchases with new version purchases
              ✓ Latest Key is owned
              ✓ Latest publicLock version is correct

  Testing version 3
Compiling 2 files with 0.5.7
contracts/past-versions/UnlockV3.sol:2491:5: Warning: Unused function parameter. Remove or comment out the
    address _purchaser, // solhint-disable-line no-unused-vars
    ^----------------^

contracts/past-versions/UnlockV3.sol:2492:5: Warning: Unused function parameter. Remove or comment out the
    uint _keyPrice // solhint-disable-line no-unused-vars
    ^------------^

contracts/past-versions/UnlockV3.sol:2512:5: Warning: Unused function parameter. Remove or comment out the
    address _referrer // solhint-disable-line no-unused-vars
    ^---------------^

contracts/past-versions/UnlockV3.sol:2530:5: Warning: Unused function parameter. Remove or comment out the
    uint _tokens // solhint-disable-line no-unused-vars
    ^----------^

Compilation finished successfully
    ✓ Unlock version is set
    ✓ this version and latest version have different Unlock bytecode
    ✓ Unlock has an owner
    Complete PublicLock configuration if require
      ✓ this version and latest version have different PublicLock bytecode
      Create a lock for testing
        ✓ PublicLock version is set
        Purchase a key
          ✓ Key has an ID
          ✓ Key is owned
          Upgrade Unlock to latest version
            ✓ this version and latest version have different Unlock version numbers
            ✓ latest version number is correct
            ✓ Key id still set
            ✓ Key is still owned
            ✓ New keys may still be purchased
            ✓ Keys may still be transferred
            ✓ grossNetworkProduct remains
            ✓ lock data should persist state between upgrades
            ✓ tokenURI still works as expected
            Using latest version after an upgrade
              ✓ this version and latest version have different PublicLock version numbers
              ✓ grossNetworkProduct sums previous version purchases with new version purchases
              ✓ Latest Key is owned
              ✓ Latest publicLock version is correct

  Testing version 4
Downloading compiler 0.5.9
Compiling 2 files with 0.5.9
contracts/past-versions/UnlockV4.sol:2643:5: Warning: Unused function parameter. Remove or comment out the
    address _purchaser, // solhint-disable-line no-unused-vars
    ^----------------^

contracts/past-versions/UnlockV4.sol:2644:5: Warning: Unused function parameter. Remove or comment out the
    uint _keyPrice // solhint-disable-line no-unused-vars
    ^------------^

contracts/past-versions/UnlockV4.sol:2664:5: Warning: Unused function parameter. Remove or comment out the
    address _referrer // solhint-disable-line no-unused-vars
    ^---------------^

contracts/past-versions/UnlockV4.sol:2682:5: Warning: Unused function parameter. Remove or comment out the
    uint _tokens // solhint-disable-line no-unused-vars
    ^----------^

Compilation finished successfully
    ✓ Unlock version is set
    ✓ this version and latest version have different Unlock bytecode
    ✓ Unlock has an owner
    Complete PublicLock configuration if require
      ✓ this version and latest version have different PublicLock bytecode
      Create a lock for testing
        ✓ PublicLock version is set
        Purchase a key
          ✓ Key has an ID
          ✓ Key is owned
          Upgrade Unlock to latest version
            ✓ this version and latest version have different Unlock version numbers
            ✓ latest version number is correct
            ✓ Key id still set
            ✓ Key is still owned
            ✓ New keys may still be purchased
            ✓ Keys may still be transferred
            ✓ grossNetworkProduct remains
            ✓ lock data should persist state between upgrades
            ✓ tokenURI still works as expected
            Using latest version after an upgrade
              ✓ this version and latest version have different PublicLock version numbers
            ✓ grossNetworkProduct sums previous version purchases with new version purchases
            ✓ Latest Key is owned
            ✓ Latest publicLock version is correct

  Testing version 6
Downloading compiler 0.5.14
Compiling 2 files with 0.5.14
contracts/past-versions/UnlockV6.sol:1162:5: Warning: Unused function parameter. Remove or comment out the
    address _purchaser, // solhint-disable-line no-unused-vars
    ^----------------^

contracts/past-versions/UnlockV6.sol:1163:5: Warning: Unused function parameter. Remove or comment out the
    uint _keyPrice // solhint-disable-line no-unused-vars
    ^------------^

contracts/past-versions/UnlockV6.sol:1183:5: Warning: Unused function parameter. Remove or comment out the
    address _referrer // solhint-disable-line no-unused-vars
    ^---------------^

contracts/past-versions/UnlockV6.sol:1219:5: Warning: Unused function parameter. Remove or comment out the
    uint _tokens // solhint-disable-line no-unused-vars
    ^----------^

Compilation finished successfully
    ✓ Unlock version is set
    ✓ this version and latest version have different Unlock bytecode
    ✓ Unlock has an owner
    Complete PublicLock configuration if require
      ✓ this version and latest version have different PublicLock bytecode
      Create a lock for testing
        ✓ PublicLock version is set
        Purchase a key
          ✓ Key has an ID
          ✓ Key is owned
          Upgrade Unlock to latest version
            ✓ this version and latest version have different Unlock version numbers
            ✓ latest version number is correct
            ✓ Key id still set
            ✓ Key is still owned
            ✓ New keys may still be purchased
            ✓ Keys may still be transferred
            ✓ grossNetworkProduct remains
            ✓ lock data should persist state between upgrades
            ✓ tokenURI still works as expected
            Using latest version after an upgrade
              ✓ this version and latest version have different PublicLock version numbers
              ✓ grossNetworkProduct sums previous version purchases with new version purchases
              ✓ Latest Key is owned
              ✓ Latest publicLock version is correct

  Testing version 7
```

```
Compiling 2 files with 0.5.17
Compilation finished successfully
    ✓ Unlock version is set
    ✓ this version and latest version have different Unlock bytecode
    ✓ Unlock has an owner
    Complete PublicLock configuration if require
      ✓ this version and latest version have different PublicLock bytecode
      Create a lock for testing
        ✓ PublicLock version is set
        Purchase a key
          ✓ Key has an ID
          ✓ Key is owned
          Upgrade Unlock to latest version
            ✓ this version and latest version have different Unlock version numbers
            ✓ latest version number is correct
            ✓ Key id still set
            ✓ Key is still owned
            ✓ New keys may still be purchased
            ✓ Keys may still be transferred
            ✓ grossNetworkProduct remains
            ✓ lock data should persist state between upgrades
            ✓ tokenURI still works as expected
            Using latest version after an upgrade
              ✓ this version and latest version have different PublicLock version numbers
              ✓ grossNetworkProduct sums previous version purchases with new version purchases
              ✓ Latest Key is owned
              ✓ Latest publicLock version is correct

  Testing version 8
Compiling 2 files with 0.5.17
Compilation finished successfully
    ✓ Unlock version is set
    ✓ this version and latest version have different Unlock bytecode
    ✓ Unlock has an owner
    Complete PublicLock configuration if require
      ✓ this version and latest version have different PublicLock bytecode
      Create a lock for testing
        ✓ PublicLock version is set
        Purchase a key
          ✓ Key has an ID
          ✓ Key is owned
          Upgrade Unlock to latest version
            ✓ this version and latest version have different Unlock version numbers
            ✓ latest version number is correct
            ✓ Key id still set
            ✓ Key is still owned
            ✓ New keys may still be purchased
            ✓ Keys may still be transferred
            ✓ grossNetworkProduct remains
            ✓ lock data should persist state between upgrades
            ✓ tokenURI still works as expected
          Using latest version after an upgrade
            ✓ this version and latest version have different PublicLock version numbers
            ✓ grossNetworkProduct sums previous version purchases with new version purchases
            ✓ Latest Key is owned
            ✓ Latest publicLock version is correct

  Testing version 9
Compiling 1 file with 0.5.17
Compiling 1 file with 0.8.4
Compilation finished successfully
    ✓ Unlock version is set
    ✓ this version and latest version have different Unlock bytecode
    ✓ Unlock has an owner
    Complete PublicLock configuration if require
      ✓ this version and latest version have different PublicLock bytecode
      Create a lock for testing
        ✓ PublicLock version is set
        Purchase a key
          ✓ Key has an ID
          ✓ Key is owned
          Upgrade Unlock to latest version
            ✓ this version and latest version have different Unlock version numbers
            ✓ latest version number is correct
            ✓ Key id still set
            ✓ Key is still owned
            ✓ New keys may still be purchased
            ✓ Keys may still be transferred
            ✓ grossNetworkProduct remains
            ✓ lock data should persist state between upgrades
            ✓ tokenURI still works as expected
            Using latest version after an upgrade
              ✓ this version and latest version have different PublicLock version numbers
              ✓ grossNetworkProduct sums previous version purchases with new version purchases
              ✓ Latest Key is owned
              ✓ Latest publicLock version is correct

·------------------------------------------------------------|---------------------------|-------------|---
|                   Solc version: 0.4.24                     ·  Optimizer enabled: true  ·  Runs: 200  ·  B
·······························································|···························|·············|···
|  Methods
··································|·························|·············|·············|·············|···
|  Contract                       ·  Method                 ·  Min        ·  Max        ·  Avg        ·  #
··································|·························|·············|·············|·············|···
|  ERC20                          ·  approve                ·      38499  ·      65796  ·      49690  ·
··································|·························|·············|·············|·············|···
|  ERC20                          ·  transfer               ·      41799  ·     141495  ·     110435  ·
··································|·························|·············|·············|·············|···
|  ERC20                          ·  transferFrom           ·      53415  ·     158052  ·     103977  ·
··································|·························|·············|·············|·············|···
|  ERC20Mintable                  ·  mint                   ·      43905  ·     127997  ·      84226  ·
··································|·························|·············|·············|·············|···
```

```
|  MinterRole                       ·  addMinter                 ·      55114  ·      55163  ·      55142  ·
·································|···························|·············|·············|·············|···
|  MinterRole                       ·  initialize                ·          -  ·          -  ·      68714  ·
·································|···························|·············|·············|·············|···
|  MinterRole                       ·  renounceMinter            ·          -  ·          -  ·      30474  ·
·································|···························|·············|·············|·············|···
|  ProxyAdmin                       ·  upgrade                   ·      38802  ·      38814  ·      38813  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  addKeyGranter             ·          -  ·          -  ·      59884  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  addLockManager            ·      50399  ·      59839  ·      57479  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  approveBeneficiary        ·          -  ·          -  ·      61522  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  cancelAndRefund           ·      57226  ·      93989  ·      73121  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  disableLock               ·          -  ·          -  ·      32091  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  expireAndRefundFor        ·      45583  ·      64996  ·      48142  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  grantKeys                 ·      47115  ·     151625  ·      89696  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  initialize                ·          -  ·          -  ·     331805  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  purchase                  ·      78628  ·     339407  ·     187309  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  renounceLockManager       ·          -  ·          -  ·      32470  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  revokeKeyGranter          ·          -  ·          -  ·      37899  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  safeTransferFrom          ·          -  ·          -  ·     111897  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  safeTransferFrom          ·          -  ·          -  ·     112702  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  setApprovalForAll         ·      33730  ·      55654  ·      47282  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  setBaseTokenURI           ·      37582  ·      99148  ·      63715  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  setEventHooks             ·      63944  ·      63956  ·      63954  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  setExpirationDuration     ·          -  ·          -  ·      36195  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  setGasRefundValue         ·          -  ·          -  ·      53374  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  setKeyManagerOf           ·      38564  ·      60965  ·      53243  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  setMaxNumberOfKeys        ·          -  ·          -  ·      38306  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  shareKey                  ·      66759  ·     139142  ·     107693  ·
·································|···························|·············|·············|·············|···
|  PublicLock                       ·  updateBeneficiary         ·      36634  ·      36774  ·      36685  ·
```

```
··································|··························|··············|··············|··············|···
|  PublicLock                      ·  updateKeyPricing         ·       38094  ·       85562  ·       58317  ·
··································|··························|··············|··············|··············|···
|  PublicLock                      ·  updateLockName           ·       34363  ·       39464  ·       38171  ·
··································|··························|··············|··············|··············|···
|  PublicLock                      ·  updateLockSymbol         ·           -  ·           -  ·       56036  ·
··································|··························|··············|··············|··············|···
|  PublicLock                      ·  updateRefundPenalty      ·       39874  ·       59786  ·       51587  ·
··································|··························|··············|··············|··············|···
|  PublicLock                      ·  updateTransferFee        ·       34429  ·       54375  ·       42701  ·
··································|··························|··············|··············|··············|···
|  PublicLock                      ·  withdraw                 ·       43167  ·       66909  ·       46476  ·
··································|··························|··············|··············|··············|···
|  TestEventHooks                  ·  configure                ·       45907  ·       66191  ·       58468  ·
··································|··························|··············|··············|··············|···
|  TestEventHooks                  ·  setSpecialMember         ·           -  ·           -  ·       44292  ·
··································|··························|··············|··············|··············|···
|  TimelockControllerUpgradeable   ·  grantRole                ·           -  ·           -  ·       58794  ·
··································|··························|··············|··············|··············|···
|  TimeMachineMock                 ·  timeMachine              ·       40886  ·       40930  ·       40905  ·
··································|··························|··············|··············|··············|···
|  Unlock                          ·  addLockTemplate          ·       39844  ·       99774  ·       98327  ·
··································|··························|··············|··············|··············|···
|  Unlock                          ·  configUnlock             ·       52799  ·      135550  ·      101563  ·
··································|··························|··············|··············|··············|···
|  Unlock                          ·  createLock               ·      418415  ·      945620  ·      456021  ·
··································|··························|··············|··············|··············|···
|  Unlock                          ·  createUpgradeableLock    ·      925858  ·      971341  ·      947147  ·
··································|··························|··············|··············|··············|···
|  Unlock                          ·  initializeProxyAdmin     ·           -  ·           -  ·      497246  ·
··································|··························|··············|··············|··············|···
|  Unlock                          ·  resetTrackedValue        ·       34903  ·       39787  ·       39154  ·
··································|··························|··············|··············|··············|···
|  Unlock                          ·  setLockTemplate          ·      289829  ·      351622  ·      340414  ·
··································|··························|··············|··············|··············|···
|  Unlock                          ·  setOracle                ·      138163  ·      138185  ·      138179  ·
··································|··························|··············|··············|··············|···
|  Unlock                          ·  upgradeLock              ·           -  ·           -  ·       65651  ·
··································|··························|··············|··············|··············|···
|  UnlockDiscountTokenV2           ·  delegate                 ·       35725  ·      102516  ·       50032  ·
··································|··························|··············|··············|··············|···
|  UnlockDiscountTokenV3           ·  delegate                 ·       48308  ·      110549  ·       86635  ·
··································|··························|··············|··············|··············|···
|  UnlockDiscountTokenV3           ·  initialize               ·           -  ·           -  ·      205618  ·
··································|··························|··············|··············|··············|···
|  UnlockDiscountTokenV3           ·  mint                     ·      104571  ·      167545  ·      125062  ·
··································|··························|··············|··············|··············|···
|  UnlockDiscountTokenV3           ·  transfer                 ·       49751  ·      135624  ·       62589  ·
··································|··························|··············|··············|··············|···
|  UnlockProtocolGovernor          ·  castVote                 ·      106194  ·      109193  ·      107194  ·
··································|··························|··············|··············|··············|···
```

```
|  UnlockProtocolGovernor          ·  execute                 ·      110963  ·      112892  ·      112226  ·
·································|·························|·············|·············|·············|···
|  UnlockProtocolGovernor          ·  propose                 ·      109206  ·      109990  ·      109471  ·
·································|·························|·············|·············|·············|···
|  UnlockProtocolGovernor          ·  queue                   ·      129345  ·      129417  ·      129373  ·
·································|·························|·············|·············|·············|···
|  Deployments                                                ·                                          ·  %
···························································|·············|·············|·············|···
|  KeyManagerMock                                             ·           -  ·           -  ·     5069910  ·
···························································|·············|·············|·············|···
|  LockSerializer                                             ·           -  ·           -  ·     1094241  ·
···························································|·············|·············|·············|···
|  PublicLock                                                 ·           -  ·           -  ·     5059041  ·
···························································|·············|·············|·············|···
|  TestEventHooks                                             ·           -  ·           -  ·      738575  ·
···························································|·············|·············|·············|···
|  TestPublicLockUpgraded                                     ·           -  ·           -  ·     5072655  ·
···························································|·············|·············|·············|···
|  TimeMachineMock                                            ·           -  ·           -  ·     5081832  ·
···························································|·············|·············|·············|···
|  Unlock                                                     ·           -  ·           -  ·     3383444  ·
···························································|·············|·············|·············|···
|  UnlockDiscountToken                                        ·           -  ·           -  ·     1170498  ·
···························································|·············|·············|·············|···
|  UnlockDiscountTokenV3                                      ·           -  ·           -  ·     2294799  ·
···························································|·············|·············|·············|···
|  UnlockProtocolGovernor                                     ·           -  ·           -  ·     2479977  ·
···························································|·············|·············|·············|···
|  UnlockProtocolTimelock                                     ·           -  ·           -  ·     1711854  ·
···························································|·············|·············|·············|···
|  UnlockUtilsMock                                            ·           -  ·           -  ·      489711  ·
·----------------------------------------------------------|-------------|-------------|-------------|---


  632 passing (10m)
  50 pending
```

License

This report falls under the terms described in the included LICENSE.