Non-Custodial Options using Elements

Sanket Kanjalkar, Allen Piscitello, Andrew Poelstra

October 20, 2022

Abstract

Elements is a blockchain platform that extends Bitcoin to allow issuing additional assets. Elements uses Bitcoin's opcodes along with additional opcodes not available in Bitcoin. The addition of the introspective opcodes in Elements allow for more complex transactions that can allow for blockchain-level enforcement of these transfers based on the construction of the transaction itself. Introspective opcodes can be used to construct covenants that can enforce numerous financial instruments, such as options contracts, without the need for a central custodian or escrow agent. This document will describe how a non-custodial option contract can be implemented using Miniscript Extensions, which utilize the additional introspective and arithmetic opcodes added to Elements.

1 Options

Options represent the rights to buy an asset (referred to as the **Collateral Asset**) from another party at some point in the future for a given price, measured in another asset (referred to as the **Settlement Asset**). The party that obtains this right (referred to as the **Option Grantee**) typically will pay a premium to the seller (referred to as the **Option Grantor**) for this privilege. The Option Grantee has the right but not obligation to exercise this option. If the option has not been exercised before an **expiry date**, the Option Grantor is able to keep their pledged collateral.

An additional variant of an option contract is the right without obligation to sell an asset to another party at some point in the future for a given price. Functionally this is identical to having an option to buy the Settlement Asset for the Collateral Asset, therefore we will always refer to options as having the rights to buy an asset, even though in practice it is thought of as the rights to sell an asset to a buyer.

Options typically come in two varieties – cash settled and physically settled. In a physically settled option, there is a delivery of the underlying collateral and settlement to each of the parties involved. In cash settled, rather than transfer the assets involved, a cash asset that represents the gain or loss of the trade, when all assets would be sold at a market price. For example, if Alice wished to sell Bob an option contract on Bitcoin with a $50,000 strike price through

cash settled options, Bob would receive USD from Alice, rather than Bitcoin, if he exercised the option. In this example, suppose the price was $60,000 at the time of exercise. Rather than receiving 1 Bitcoin from Alice in exchange for $50,000, Alice would transfer Bob $10,000, the net of the $60,000 value of Bitcoin and the $50,000 strike price. Cash settled options can often times have no underlying collateral backing the contracts, and in extreme price swings, can leave the option seller unable to fulfill their obligation.

Options are typically secured through legal contracts or through custodians. If the Option Grantee wishes to execute the option and the Option Grantor is unable or unwilling to fulfill this obligation, the conflict may be resolved in court, typically at great expense to both parties. An Option Grantor additionally may not be able to fulfill their obligation due to lack of liquidity or assets, leaving the Option Grantee with little recourse. An alternative approach is to require the Option Grantor to transfer their Collateral Asset to a third party to hold in escrow, ensuring that the collateral will be available for the Grantee at the expiry date. This also contains risk – the escrow agent now must keep this asset secure, and must be trusted to fulfill the obligation on behalf of the Option Grantee.

Smart Contracts allow the rules of the Option to be encoded into code such that no third party is needed to enforce the contract other than the enforcement of the rules native to a blockchain.

Elements has sufficient power to create and enforce options contracts for assets on the Elements platform. Physically settled options contracts remove the need for any third party price oracles.

2 Elements Introspection Opcodes

Elements has introduced several opcodes that allow for the introspection of the transaction contents. These introspection opcodes allow for inspecting all parts of the transaction such as the asset, value, and script of each input and output. Additionally, 64-bit signed arithmetic operations have been added to allow for calculations to be made over the values of the Elements transactions.

2.1 Limit Order Example

A limit order is an offer to sell or buy an asset for a given price, up to a maximum quantity. Limit orders can be enforced on Elements using the Introspection Opcodes and the signed 64-bit arithmetic opcodes. For example, if a trader wishes to sell up to 5 L-BTC in exchange for 20,000 USDt per L-BTC, the seller can encumber 5 L-BTC into a covenant that allows any buyer to remove up to 5 L-BTC from the covenant as long as they send 20,000 USDt per L-BTC removed to an address specified by the seller. If the buyer wishes to trade less than the 5 L-BTC limit, they simply create a change output that also is encumbered by the same covenant of the amount remaining. In this case, the seller is called the

maker and the buyer is considered the taker of the limit order. This allows for partial fills of the limit orders. The covenant would have the following rules:

- Input 0 is the covenant being spent. This ensures that multiple covenants cannot re-use the same change output when calculating how much has been removed from the covenant.
- Output 0 is the amount paid to the limit order maker.
  - The Asset must be explicit and must be USDt.
  - The Script must be the seller's specified address.
- Output 1 is any change returned to the covenant, if change is needed. If there is any change remaining (20000 $\times$ [amount of Input 0] $>$ [amount of Output 0]), there must be a change output:
  - The asset must be explicit and must be the same as Input 0.
  - The change must be the correct amount: [amount of Output 0] = ([amount of Output 1] - [amount of Input 0]) $\times$ 20000.
  - The Script must be equal to the Script of the output spent at Input 0.

2.2 Covenant Example

In this example, Alice funds a Covenant with 5 L-BTC. This Covenant allows for the removal of 1 L-BTC per 20,000 USDt sent to Alice's Wallet. Bob wishes to trade against this order, but only wishes to purchase 2 of the 5 L-BTC available. Bob constructs a transaction that spends the covenant's UTXO in Input 0 and adds a UTXO that allows him to send 40,000 USDt. He adds outputs that pay Alice her desired amount of 40,000 USDt and returns 3 L-BTC back to the covenant. Bob is able to claim the 2 L-BTC (minus transaction fees) and 10,000 USDt as change.

3 Miniscript Extensions

In 2020, Blockstream research developed Miniscript, a new way to model Bitcoin Script that would allow generic signing, semantic analysis and composition.

With Miniscript, it is possible to automatically enumerate all the keys in a script and determine which sets are needed to produce a transaction; find an upper bound on the size of such a transaction prior to gathering signatures; answer semantic questions about which conditions must be met in order to spend coins; and compute the exact encodings for scripts and witnesses.

We have written an alternate version of Miniscript, extended to use the new opcodes in Elements, to facilitate creation of more complex covenant scripts. To this end, we added "extensions" to Miniscript that allows interacting with these new opcodes. Broadly, put the new extensions can be categorized into two types:

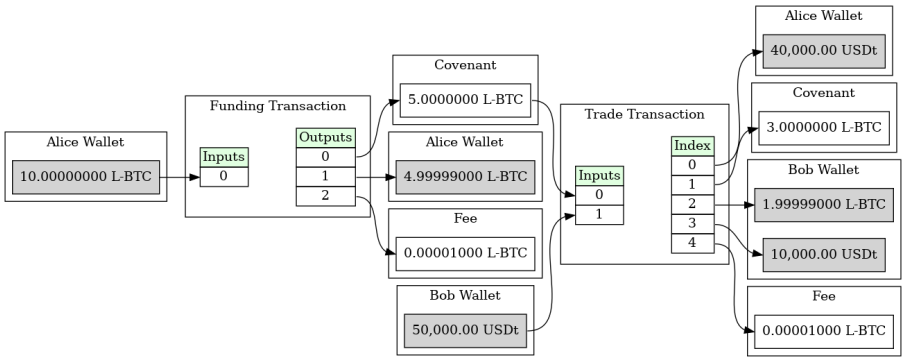

Figure 1: A transaction graph featuring a limit order covenant. All confidential outputs are shaded.

- Extensions that allow introspection of the transaction data. These include support for introspection of the transaction data, such as the asset, value, and script of each input and output. For example,

  asset_eq(out_asset(0),USDT_ASSET) can only be satisfied if the asset of the first output is equal to the USDT_ASSET.

- Extensions that allow arithmetic 64 bit operations like comparisons, arithmetic operations, and logical operations. For example,

  num64_gt(out_v(0),100) can only be satisfied if the value of the first output is greater than 100.

There are new extensions that also support EC operations and streaming hash operations. Those are explained in more detail in the extension spec[1].

As with regular Miniscript, these extension fragments can be composed with **and**, **or** and **thresh** to generate more complicated conditions. As a small example, using the above constraints we can create the following script

- s1 = spk_eq(out_spk(0), OP_RETURN)
- s2 = value_eq(out_v(0), 1)
- s3 = asset_eq(out_asset(0), BURN_ASSET)

And finally combine them using thresh(3,s1,s2,s3). This gives us a systematic and easy way to write a script which can only be unlocked if the spending transaction output 0 spends 1 sat of BURN_ASSET to OP_RETURN address.

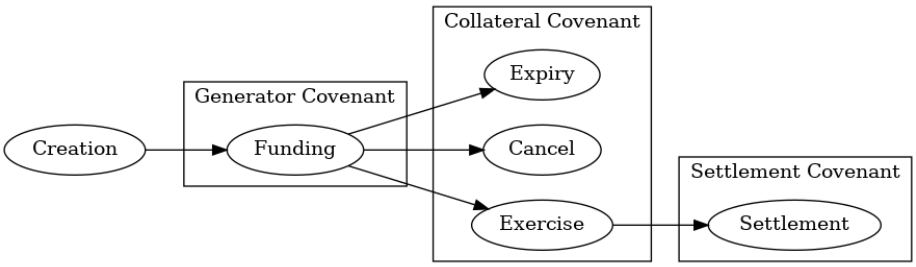

Figure 2: The Options Lifecycle

4 Options Workflow

An option first must be defined and created. The option creator must specify the parameters of the option contract, such as the Settlement and Collateral Assets, the strike price and contract size, and expiry date. An option can then be generated from these parameters. The result of this process is a set of Generators that allow for the creation of options contracts. No contracts have been issued at this point.

Options can be funded once they have been created. Funding an option consists of creating a transaction that adds collateral into a covenant. The funder of an option is then provided with a pair of tokens that represent the two parties in an option – the grantee and grantor. The grantee is represented through an **Option Token**, which gives the holder the right but not obligation to purchase the collateral at a future date. The grantor is represented through a **Grantor Token**, which gives the holder the ability to reclaim the collateral after the contract has expired without being executed or the right to claim the settlement payment made by the grantee. Once created, the funder can trade these tokens with other users, depending on which position they would like to take. A funder can take three positions – either as a market maker who tries to trade the tokens to other users with a spread and profit from these discrepancies, as an option seller who wishes to get paid a premium for being willing to sell the collateral if its value increases, or as an option buyer who wishes to speculate on the price of the collateral rising in the future while maintaining little downside exposure. Each token generated represents one **Contract Size** amount of collateral. If a user funds larger than this amount, they are compensated with multiple tokens.

An option contract can then follow three distinct paths. The first path is when the contract expires without being executed. The holder of a Grantor token can redeem this token for the corresponding collateral after the expiry date. The second path allows for a user to redeem a pair of Grantor and Option Tokens for the corresponding collateral. This allows for early access to the collateral by holding offsetting tokens. The final path is exercising the option.

[1]The detailed spec can be found at https://github.com/ElementsProject/elements-miniscript/blob/6029aba1fc31323877276e6e3d1311c25a519d3c/doc/extension_spec.md

Exercising an option can happen at any time after a defined start date, and as long as the underlying collateral has not been claimed by a holder of the Grantor token. While this does mean that options can be exercised after their expiry date, the Grantor has an incentive to claim the underlying collateral as early as possible. The holder redeems the Option Token for the underlying collateral by sending the payment to a Settlement Covenant, to be claimed by an Options Grantor. It is possible for a single option contract to have all three paths followed during its lifetime, however it is likely that all users will either follow only one of the Expiry or Exercise paths. The cancellation path is likely one that can be used by market makers to shrink liquidity when needed.

After an option has been exercised, the holder of a Grantor Token can redeem this token for the Settlement Asset. In the event that not all options of the same type have been exercised, the Grantor token holder can choose, from the collateral available, if they prefer claiming the Settlement Asset or Collateral Asset.

4.1 Options Creation

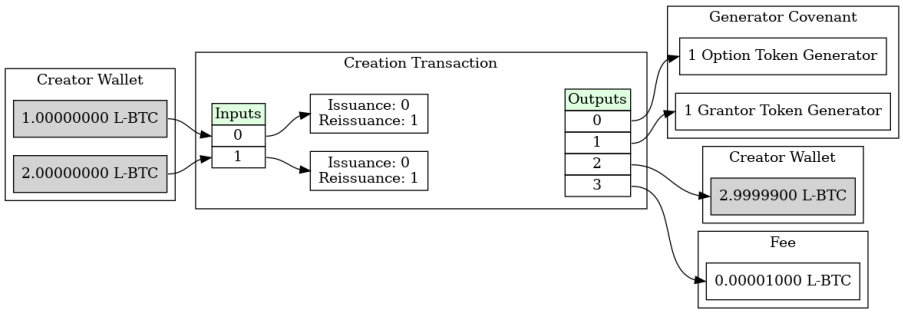

Figure 3: An Option Creation Transaction. All confidential outputs are shaded.

The parameters of the options contract is defined by six parameters:

- **Collateral Asset** – The asset that is locked in a covenant that a **Grantee** is able to purchase for a set price.
- **Settlement Asset** – The asset that the purchase of the **Collateral Asset** is denominated.
- **Contract Size** – The minimum amount of a **Collateral Asset** that can be granted as an option.
- **Strike Price** – The amount per Contract Size of the **Collateral Asset** that must be paid in the Settlement Asset.
- **Execution Start Date** – The date at which an option can be executed by the Grantee. This parameter is optional.

- **Expiry Date** – The date at which the Grantor can reclaim their collateral.

Creating an option results in two Reissuance Tokens created and sent to covenants (referred to as **Generator Covenants**). Generator Covenants allow for the re-issuance of an asset given that collateral has been properly transferred into a separate covenant (referred to as the **Collateral Covenant**). Initially, there are no Option or Grantor tokens created.

This process allows for a demand-based supply of any option. Any user, at any point in time, without permission, can add to the liquidity of an option contract. Users can also remove liquidity through the cancellation operation by burning an equivalent amount of Option and Grantor tokens.

4.2 Option Funding

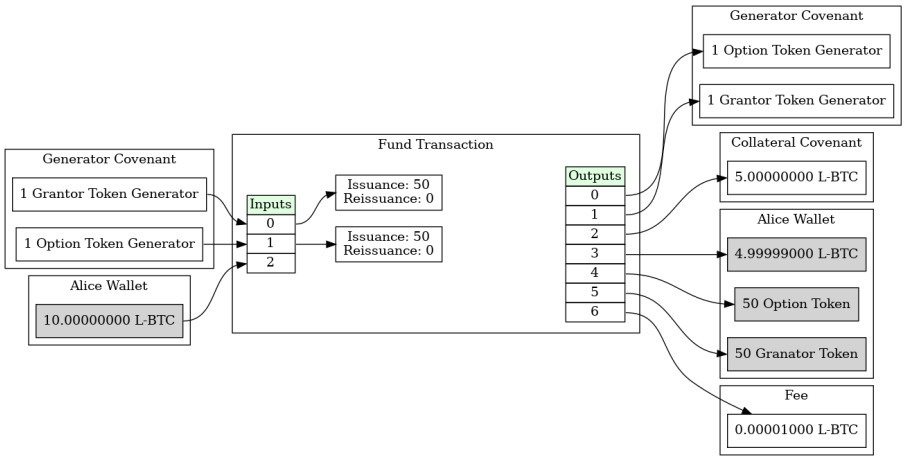

Figure 4: An Option Funding Transaction

The Generator Covenant allows the creation of a pair of Option Tokens and Grantor Tokens for every contract-size unit of the Collateral Asset transferred into the Collateral Covenant.

In this example there is an option contract with contract size of 0.1 L-BTC and a strike price of 50,000 USDt. Alice adds 5.0 L-BTC to the Collateral Covenant and is able to generate 50 new Option Tokens and 50 new Grantor Tokens. This process can be repeated multiple times by users in succession, creating as many Option Tokens and Grantee tokens as necessary, by as many parties who wish to participate in this process.

4.3 Option Trading

The party that funded the option can sell any of the tokens generated, allowing them to take a position. One way this can be done is through an atomic swap

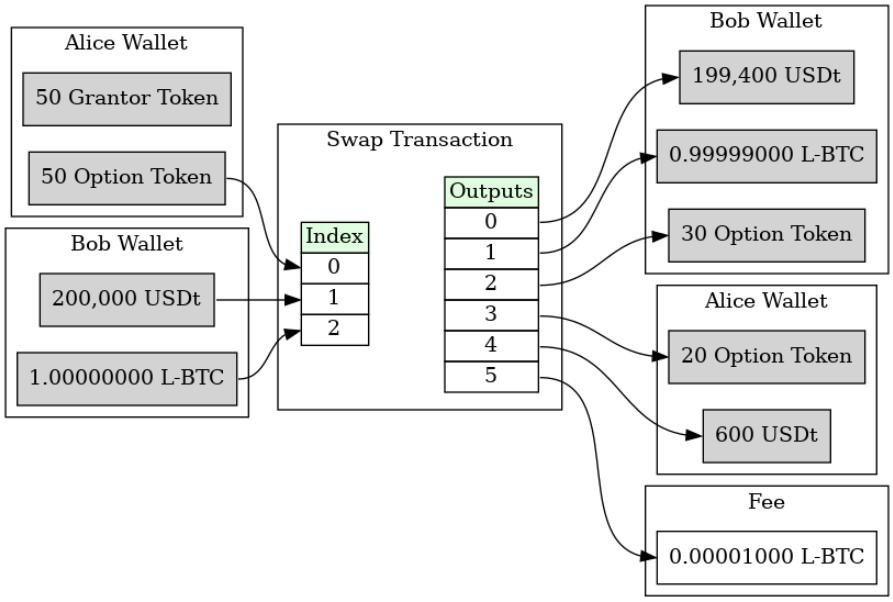

Figure 5: An Option Trading Transaction. All confidential outputs are shaded.

– an interactive trade that allows multiple parties to perform a trade without giving up control until their demands are fulfilled. These assets are unencumbered by covenants and can be freely traded or transferred as needed. Atomic swaps are only one way this can be performed.

In this example, Alice wishes to grant an option to Bob, at the price of 20 USDt per contract for 30 Option Tokens. Alice and Bob collaboratively construct a transaction that allows for this swap to occur – Alice receives net 600 USDt and Bob receives a net 30 Option Tokens. Alice retains all 50 Grantor tokens.

4.4 Option Cancellation

The Collateral Covenant allows for retrieving the collateral by redeeming a pair of Option and Grantor Tokens. For each pair redeemed, the covenant allows the removal of one contract-size amount of the Collateral Asset.

In this example, Alice was unable to sell 20 of the Option Tokens she funded. Since she is unable to find a buyer, she will redeem these 20 Option Tokens along with 20 of her Grantor Tokens in exchange for 2 L-BTC (minus transaction fees). These 20 pairs of tokens are burned, reducing the supply. After this operation, there are 30 pairs of tokens that remain and 3.0 L-BTC remaining (corresponding to 30 contracts).

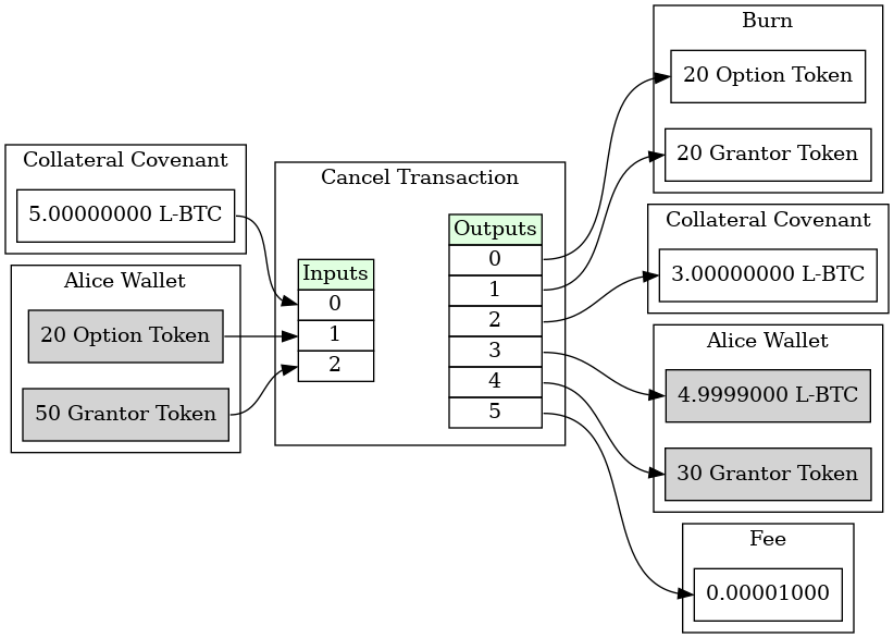

Figure 6: An Option Cancellation Transaction. All confidential outputs are shaded.

4.5 Option Exercise

The Collateral Covenant allows for redeeming an Option Token along with transferring a strike-price amount per token redeemed of the Settlement Asset into the Settlement Covenant. This allows the user to remove a contract-size amount of Collateral Asset at any point after the execution start date (if one is defined).

In this example, Bob purchased 30 Option Tokens from Alice. Bob has decided to exercise 20 of these Options by redeeming the Options Tokens and paying the Settlement Covenant 100,000 USDt. Bob cannot pay Alice directly, as Alice may have transferred her Grantor tokens to others. This also allows for fungibility between similar options contracts no matter who funded them.

4.6 Settlement Claim

The Settlement Covenant allows for redeeming a Grantor Token in exchange for one strike-price amount of the Settlement Asset.

In this example, Bob exercised 20 option contracts, depositing 100,000 USDt into the Settlement Covenant. Alice is able to redeem up to 20 Grantee Tokens by burning them for up to 100,000 USDt.

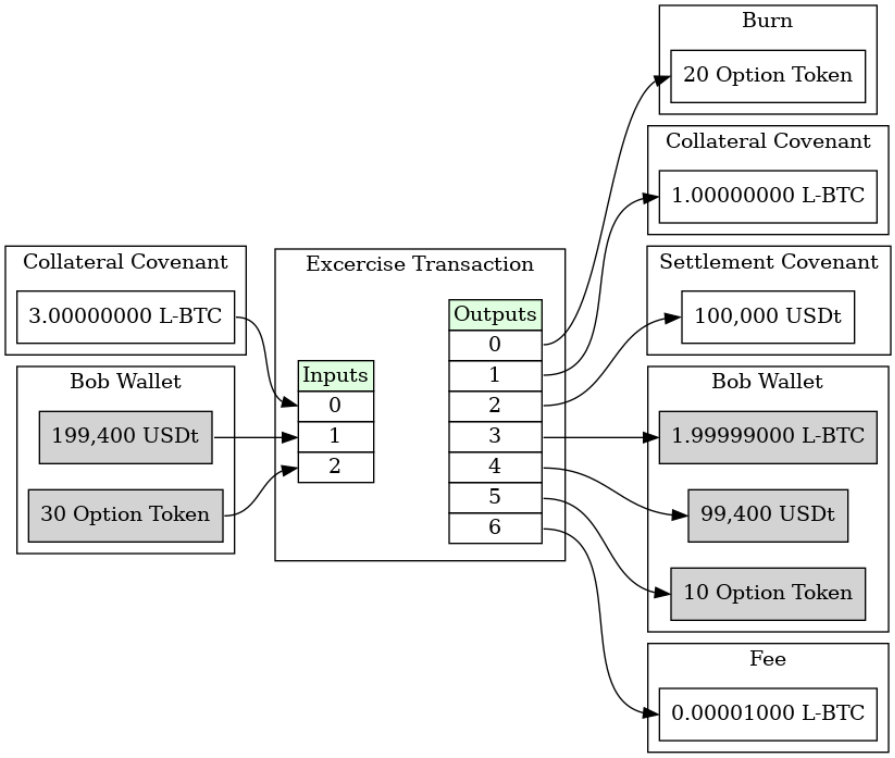

Figure 7: An Option Exercise Transaction. All confidential outputs are shaded.

4.7 Option Expiry

The Collateral Covenant allows for retrieving any unclaimed collateral by redeeming a Grantor Token after the expiry date of the option.

In this example, Bob only exercised 20 option contracts, leaving 10 unexercised. This allows Alice to claim back 10 contracts worth of collateral, equivalent to 1 L-BTC in this case. She does this by burning 10 Grantor tokens. A this point, no further Collateral and Settlement Assets remain in the covenant. Bob's unexercised Option Tokens can be discarded.

5 Covenant Definition

There are three covenants used to create options: the Generator Covenant – used to fund and generate Grantor and Collateral Tokens, the Collateral Covenant – used to manage the Collateral Asset, and the Settlement Covenant – used to manage recovery of the Settlement Asset paid during exercise of options.

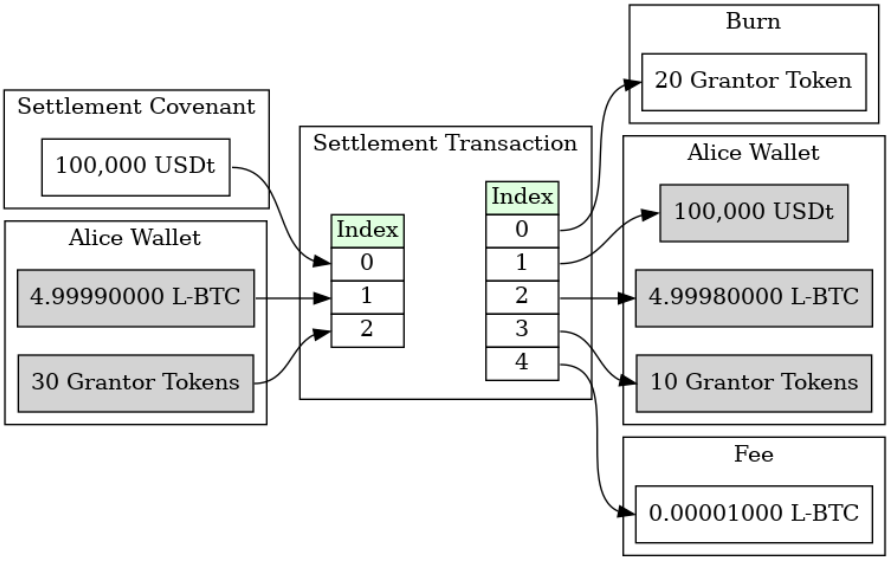

Figure 8: An Option Settlement Transaction. All confidential outputs are shaded.

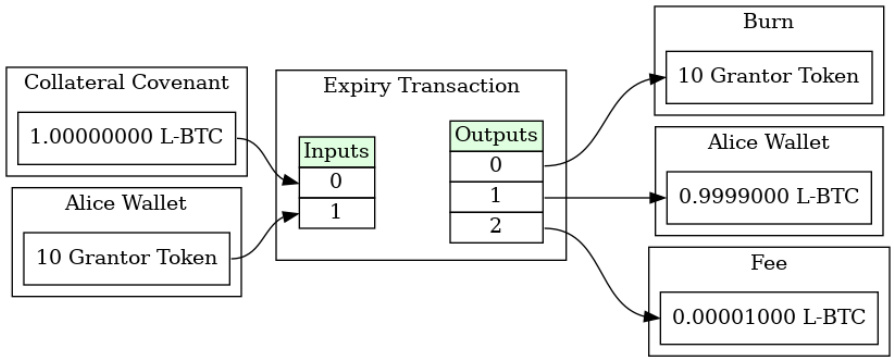

Figure 9: An Option Expiry Transaction. All confidential outputs are shaded.

5.1 Generator Covenants

The Generator Covenants encumber the two reissuance tokens (Generators) created from the Creation step. All reissuance tokens must be blinded when spent in Elements when reissuing an asset. This presents a small complication in the Generator Covenants – as it must verify the asset of the reissuance tokens in both inputs and outputs. This is worked around by using fixed asset blinding factors, alternating between 1 and 2 to create an asset commitment, but using

explicit values. This covenant can be used for both the Grantor Token Generator and the Option Token Generator.

It has the following conditions enforced:

- Input 0 is the Grantor Token Generator
  - (Rule 1) [Amount Issued] $\times$ [Contract Size] = [Output 2's Amount]
- Input 1 is the Option Token Generator
  - (Rule 1) [Amount Issued] $\times$ [Contract Size] = [Output 2's Amount]
- Output 0 is the Grantor Token Generator
  - (Rule 3) The Asset must be a commitment equal to either the Grantor Token Generator's Commitment with an Asset Blinding Factor of 1, or the Grantor Token Generator's Commitment with an Asset Blinding Factor of 2
  - (Rule 4) The Amount must be explicit and equal to the amount of Input 0
  - (Rule 5) The Script must be equal to the Script of output spent by Input 0
- Output 1 is the Option Token Generator
  - (Rule 6) The Asset must be a commitment equal to either the Option Token Generator's Commitment with an Asset Blinding Factor of 1, or the Option Token Generator's Commitment with an Asset Blinding Factor of 2
  - (Rule 7) The Amount must be explicit and equal to the amount of Input 1
  - (Rule 8) The Script must be equal to the Script of output spent by Input 1
- Output 2 is the Collateral Covenant
  - (Rule 9) The Asset must be explicit and the Collateral Asset
  - (Rule 10) The Script must be equal to the Collateral Covenant's Script
  - (Rule 2) The Amount must be equal to [Input 0 (and Input 1's)'s Amount Issued] $\times$ [Contract Size]
- (Rule 11) The current covenant must be spent from Input 0 (for the Grantor Token Generator Covenant) or from Input 1 (for the Option Token Generator Covenant)

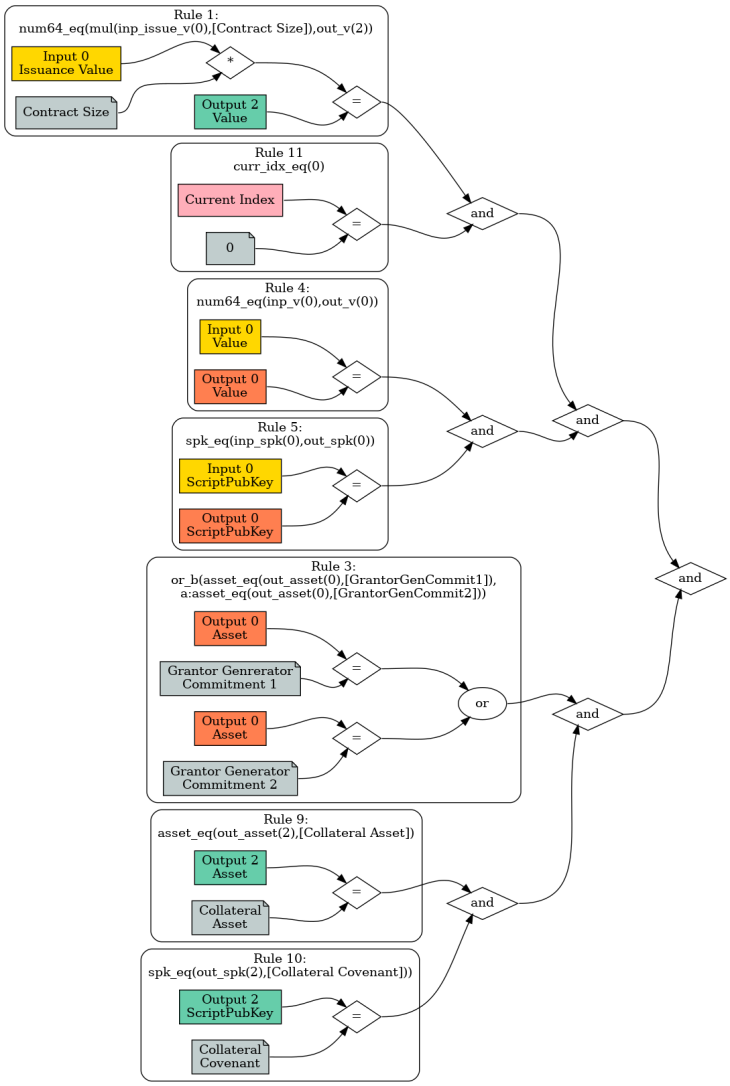

Figure 10: A graphical representation of the Miniscript used for the Grantor Token Generator Covenant

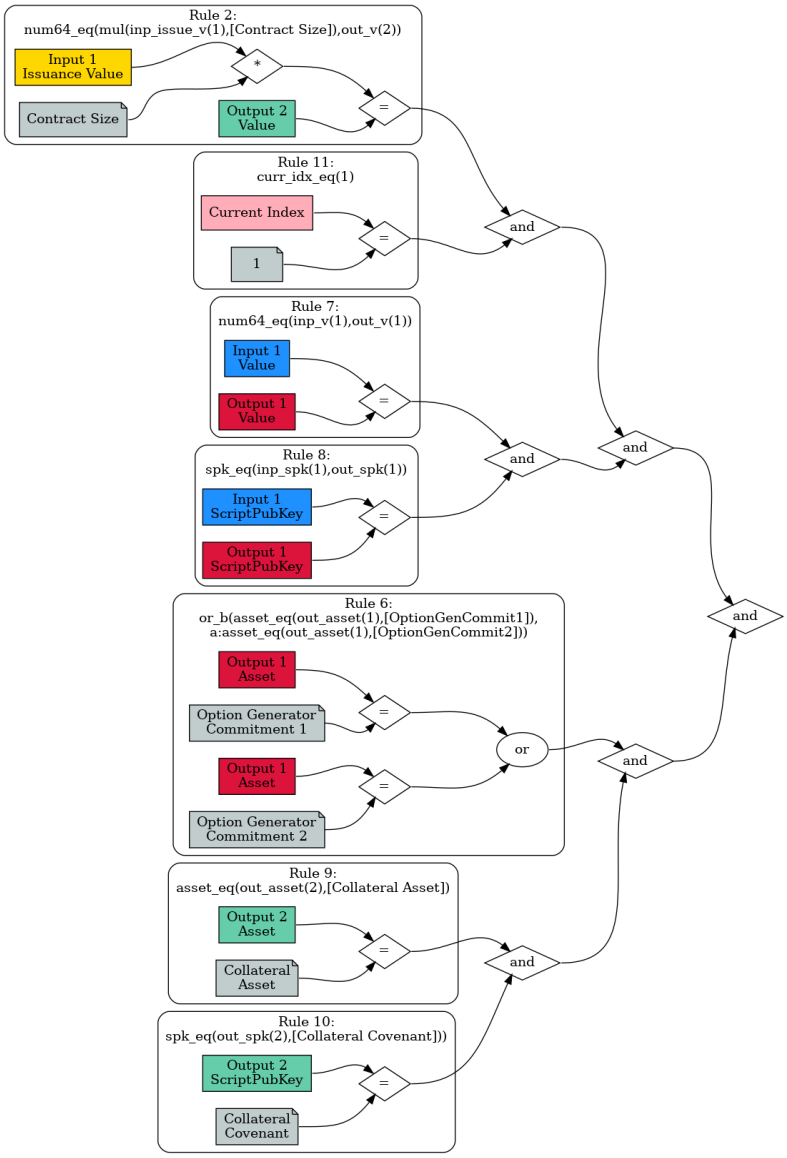

Figure 11: A graphical representation of the Miniscript used for the Option Token Generator Covenant

5.2 Collateral Covenant

The Collateral Covenant encumbers the Collateral Asset that has been pledged to options contracts during the Funding step. There are three execution paths for these cases. These execution paths are separate branches in a MAST.

5.2.1 Cancellation Path

The Cancellation Path ensures that a pair of tokens is burned for each Contract Size amount of the Collateral Asset removed from the Collateral Covenant. The Cancellation Path has the following conditions enforced:

- Input 0 is the Collateral Covenant
- Output 0 is the Option Token Burn
  - (Rule 1) The Asset must be the Option Token
  - (Rule 2) The Script must be equal to Output 1 and OP RETURN
  - (Rule 3) The Amount must be equal to Output 1's Amount
- Output 1 is the Grantor Token Burn
  - (Rule 4) The Asset must be the Grantor Token
  - (Rule 2) The Script must be equal to OP RETURN
  - (Rule 3) The Amount must be equal to Output 1's Amount
- Output 2 is the Collateral Covenant Change, if there is change.
  - (Rule 5) [Output 0's Amount] $\times$ [Contract Size] = [Input 0's Amount] **or**
  - (Rule 6) The Asset must be the Current Input's Asset
  - (Rule 7) The Script must be equal to the output spent by the Current Input's Script
  - (Rule 8) [Output 0's Amount] $\times$ [Contract Size] = [Input 0's Amount] - [Output 2's Amount]
- (Rule 9) The current covenant must be spent from Input 0.

5.2.2 Expiry Path

The Expiry Path allows burning the Grantor Token after the Expiry Date to redeem a Contract Size amount of the Collateral Asset from the Collateral Covenant. The Expiry Path has the following conditions enforced:

- Input 0 is the Collateral Covenant
- Output 0 is the Grantor Token Burn

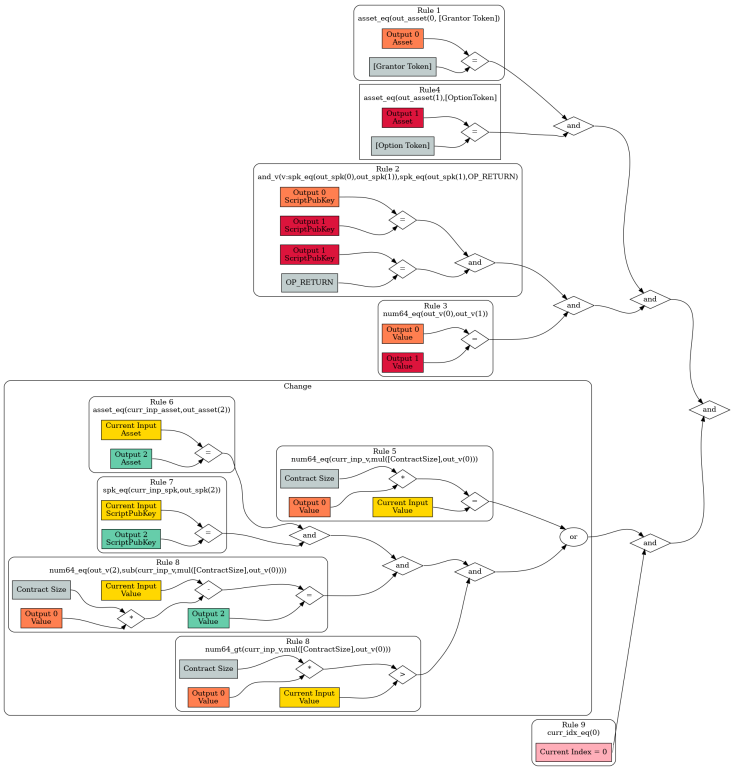

Figure 12: A graphical representation of the Miniscript used for the Collateral Covenant's Cancel Path

  - (Rule 1) The Asset must be the Grantor Token
  - (Rule 2) The Script must be equal to OP RETURN
- Output 1 is the Collateral Covenant Change, if there is change
  - (Rule 3) [Output 0's Amount] × [Contract Size] = [Input 0's Amount] **or**
  - (Rule 4) The Asset must be the Current Input's Asset
  - (Rule 5) The Script must equal be the Script of the output spent by the Current Input
  - (Rule 6) [Output 0's Amount] × [Contract Size] = [Input 0's Amount] - [Output 2's Amount]
- (Rule 7) The current covenant must be spent from Input 0.
- (Rule 8) This transaction can only occur in a block with a timestamp after the [ExpiryDate]

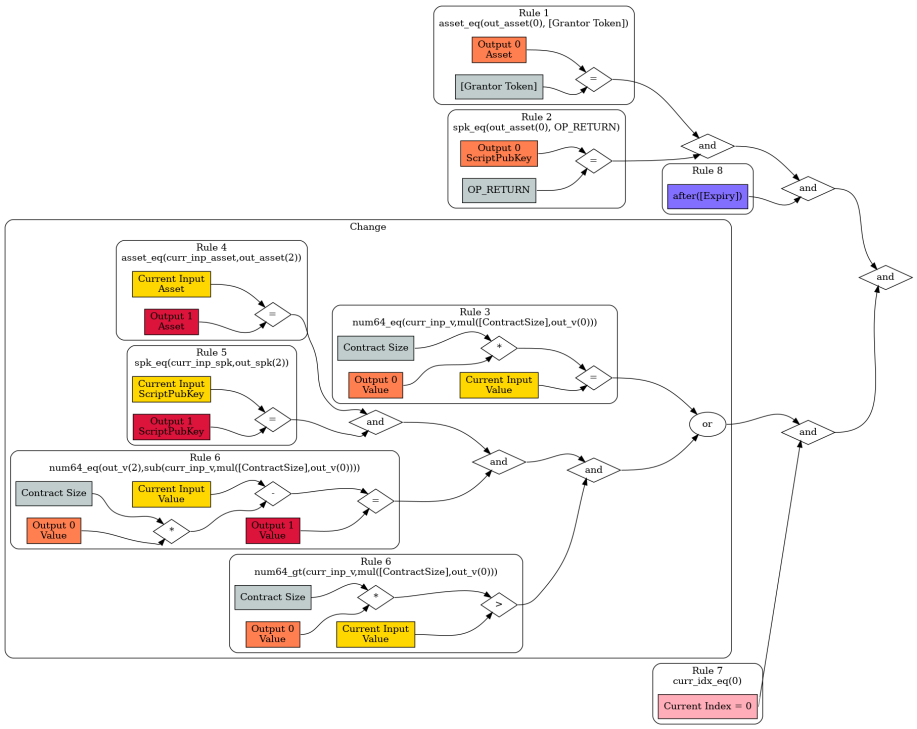

Figure 13: A graphical representation of the Miniscript used for the Collateral Covenant's Expiry Path

5.2.3 Exercise Path

The Exercise Path allows a user to burn an Option Token to retrieve a Contract Sized amount of the Collateral Asset from the Collateral Covenant, provided they send a Strike Price amount of the Settlement Asset to the Settlement Covenant. The Exercise Path has the following conditions enforced:

- Input 0 is the Collateral Covenant
- Output 0 is the Option Token Burn
  - (Rule 1) The Asset must be the Option Token
  - (Rule 2) The Script must be equal to OP RETURN
- Output 1 is the Settlement Covenant
  - (Rule 3) The Asset must be the Settlement Asset
  - (Rule 4) The Amount must be equal to [Output 0's Amount] $\times$ [Strike Price]
  - (Rule 5) The Script must be equal to the Settlement Covenant's Script
- Output 2 is the Collateral Covenant Change, if there is change
  - (Rule 6) [Output 0's Amount] $\times$ [Contract Size] = [Input 0's Amount] **or**
  - (Rule 7) The Asset must be the Current Input's Asset
  - (Rule 8) The Script must equal be the Script of the output spent by the Current Input
  - (Rule 9) [Output 0's Amount] $\times$ [Contract Size] = [Input 0's Amount] - [Output 2's Amount]
- (Rule 10) The current covenant must be spent from Input 0.
- (Rule 11) This transaction can only occur in a block with a timestamp after the [StartDate]

5.3 Settlement Covenant

The Settlement Covenant allows a user to burn a Grantor Token to retrieve a Strike Price amount of the Settlement Asset. The Settlement Covenant has the following conditions enforced:

- Input 0 is the Settlement Covenant
- Output 0 is the Grantor Token Burn
  - (Rule 1) The Asset must be the Grantor Token

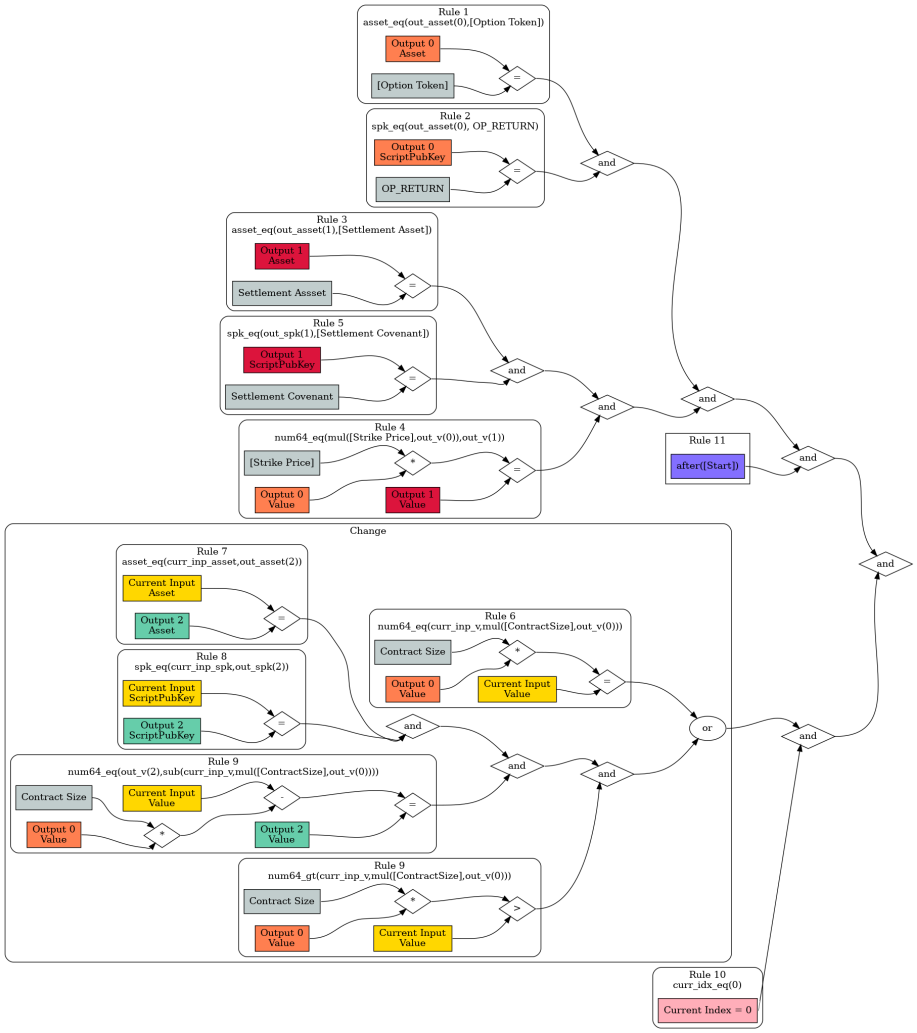

Figure 14: A graphical representation of the Miniscript used for the Collateral Covenant's Exercise Path

  - (Rule 2) The Script must be equal to OP RETURN

- Output 2 is the Settlement Covenant Change, if there is change
  - (Rule 3) [Output 0's Amount] × [Strike Price] = [Input 0's Amount] **or**
  - (Rule 4) The Asset must be the Current Input's Asset
  - (Rule 5) The Script must equal be the Script of the output spent by the Current Input
  - (Rule 6) [Output 0's Amount] × [Strike Price] = [Input 0's Amount] - [Output 2's Amount]
- (Rule 7) The current covenant must be spent from Input 0.

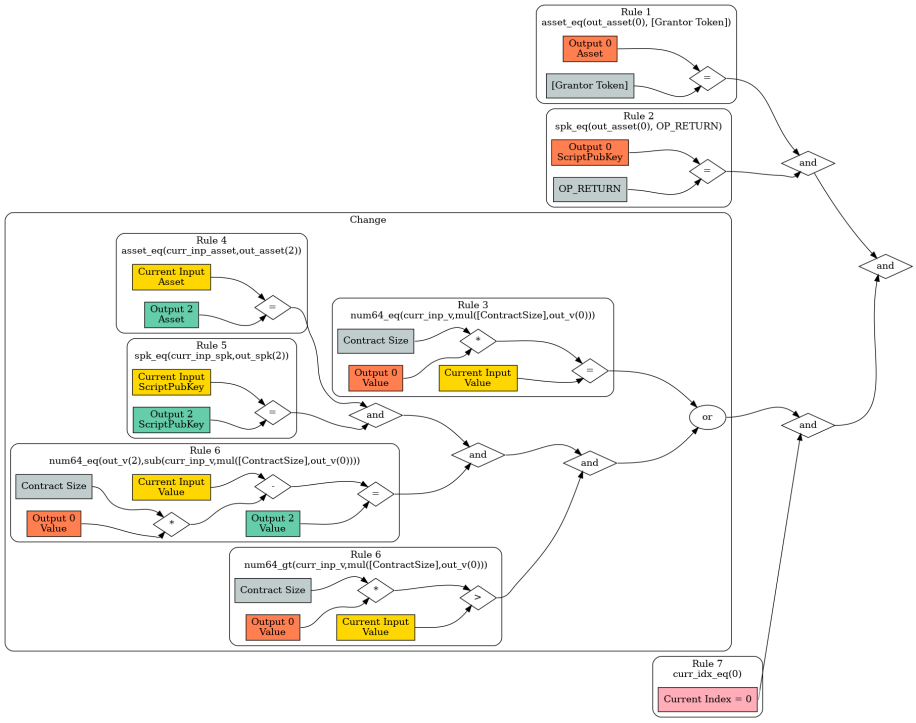

Figure 15: A graphical representation of the Miniscript used for the Settlement Covenant

6 Limitations and Risks

6.1 Bugs

Software bugs within Elements as well as the Covenants, software to create the Covenants, or other components can lead to loss of funds, either through improper access to claim funds that are not designed to be removed from a Covenant or by locking funds in Covenants without the ability to claim such funds.

6.2 Key Loss

Any access to the Collateral or Settlement Assets requires holding tokens secured by a private key. Any loss of these keys without backups would leave the rights to the option or grantor rights inaccessible without any recovery mechanism.

6.3 User Error

It is possible to put funds inside covenants without claiming the proper amount of value, such as funding a covenant without claiming tokens.

6.4 Network Failure

Any blockchain that enforces these rules could fail due to failure of functionaries, lack of interest from operators, or software bugs, which could leave funds inaccessible.

6.5 Asset Risk

There is no guarantee that the Collateral or Settlement Assets will maintain any expected value in the event of failures, such as a failure to maintain the peg of L-BTC, or through the failure of an issuer of an asset to maintain any promises. Inflation in the supply of any asset due to a software bug or key mismanagement could lead to an asset having significantly less value than expected and users must account for this in their risk.

6.6 Partial Exercising

It is possible that an in-the-money option is not fully exercised by the holders of the Option Token. It also may be possible that an out-of-the-money option is exercised by a subset of the Option Token holders. In these cases, the holders of the Grantor Tokens have a choice to try to claim the Settlement Asset or Collateral Asset, however only a small subset will be able to get the more profitable choice. There may be a bidding war with higher fees or an incentive for functionaries to favor certain parties to be able to claim the more profitable choice.

6.7 Network Censorship

Liquid relies on a round-robin consensus algorithm based on a set of M functionaries. Each of the M functionaries alternate in a set order to propose blocks, including valid transactions into the blockchain, and must receive signatures on these blocks by at least $\frac{2}{3}$ of the functionaries (including their own). A minority of $\frac{1}{3}$ of the functionaries can prevent any block that is proposed from gathering enough signatures to be considered valid. A group of functionaries that wish to prevent a party from exercising a particular option could refuse to sign any block that contains an exercise transaction. Since the introspection opcodes have many parts that are not confidential, it is immediately known to any observer that a proposed transaction is a specific option operation, such as exercising the option. One particular attack is when a set of functionaries hold an out-of-the money position, such as a Grantor token when the price of the Collateral Asset is above the Strike Price. In this case, the colluding minority could prevent any block that includes a transaction that exercises the option from being included. If $N < \frac{1}{3}$, the blockchain will continue to operate normally although they can delay transactions until a non-censoring functionary is the round leader. If $N \geq \frac{1}{3}$, blocks will only be able to be produced by the colluding N functionaries. This attack is easier to perform than blocking other transactions since the introspection opcodes used require transactions to be unblinded, thus it becomes possible for functionaries to know the purpose of a transaction.