Haven Protocol collateral notes

Cypher Stack[*]

December 19, 2022

1 Introduction

Haven Protocol recently proposed a modified transaction protocol design that introduces a collateral system. The use of collateral is intended to ensure that transactions converting between `XHV` and `xUSD` asset types produce a separate amount of `XHV` whose value depends on the converted amount present in the transaction. This approach requires modification of transaction algorithms relating to generation and verification.

We note that the calculation of the collateral amount is nontrivial, and defer an analysis of the economic implications of the calculation and collateral methodology to other work by suitable domain experts. In this technical note, we are most interested in an analysis of the modifications to transaction generation and verification as scoped by Haven Protocol.

The notes provided here reflect the codebase at the time of review[1], as well as technical notes provided by Haven describing the intent of the design.

2 Scope

Haven Protocol initially provided a specific and limited scope of code review, reflecting the following:

[*]`https://cypherstack.com`

[1]`https://github.com/haven-protocol-org/haven-main/tree/6def97c4bdee0750d2f50f3aa61cc25e2128ae67`

- `src/cryptonote_core/cryptonote_tx_utils.cpp#L721-L849`
- `src/ringct/rctSigs.cpp#L984-L994`
- `src/ringct/rctSigs.cpp#L1127-L1132`
- `src/ringct/rctSigs.cpp#L1199-1213`
- `src/ringct/rctSigs.cpp#L1487-1514`
- `src/ringct/rctSigs.cpp#L1596-1611`
- `src/cryptonote_core/tx_pool.cpp#L322-L352`
- `src/cryptonote_core/tx_pool.cpp#L424-L451`
- `src/cryptonote_core/tx_pool.cpp#L2565-L2572`
- `src/wallet/wallet2.cpp#L9854-L9859`

We again note that due to limits of domain expertise, we exclude the collateral computation from the original scope.

Haven Protocol then provided several iterations of an internal technical note describing some changes to the transaction protocol, with the intent that the scoped code matches these changes. We reviewed these as part of the analysis. Because these changes are important, we provide a modified version of them as part of this note.

3 Protocol changes

For brevity, we assume the reader is familiar with the notation from Cypher Stack's previous Haven protocol review[2], which we use here.

Haven Protocol considers changes to the *offshore* and *onshore* conversion transaction types. Offshore transactions previously consumed `XHV` outputs to generate `XHV` and `xUSD` outputs using an oracle-provided conversion rate. Onshore transactions previously consumed `xUSD` outputs to generate `xUSD` and `XHV` outputs using an oracle-provided conversion rate.

The proposed modifications introduce two new oracles. The oracle $O_{\mathtt{off}}$ takes as input an amount in `XHV` units and returns an amount in `XHV` units representing the extra collateral required to be generated in an offshore transaction. The oracle $O_{\mathtt{on}}$ takes as input an amount in `xUSD` units and returns an amount in `XHV` units representing the extra collateral required to be generated in an onshore transaction.

[2]`https://github.com/cypherstack/haven-protocol-review`

3.1 Offshore transactions

Offshore transactions are modified to require that an additional distinguished XHV output, called a *collateral* output, be generated in the transaction. The value of the collateral output depends on the result of an oracle query on the value converted in the transaction.

Let $\{v_u\}_{u=0}^{w-1}$ be the values of consumed XHV outputs in an offshore transaction. Let $\{\overline{v}_j\}_{j=0}^{n_T-1}$ be the values of XHV outputs generated. Let $\{\overline{v}_j\}_{j=n_T}^{t-1}$ be the values of xUSD outputs generated.

The transaction is modified to require that a distinguished XHV collateral output with value

$$\overline{v}_c = O_{\mathtt{off}}\left(\frac{1}{r}\sum_{j=n_T}^{t-1}\overline{v}_j\right)$$

be generated and included in the transaction. When generating this output, the sender selects a commitment mask $\overline{x}_c$ uniformly at random and defines the output as $\overline{C}_c = \mathsf{Com}(\overline{v}_c, \overline{x}_c)$. The sender includes this mask in the transaction data, and includes the collateral output when producing an aggregated range proof for the transaction.

When producing masks for the consumed output commitment offsets, the sender accounts for this new output by setting the final mask as follows:

$$x'_{w-1} = \sum_{j=0}^{n_T-1}\overline{x}_j + \frac{1}{r}\sum_{j=n_T}^{t-1}\overline{x}_j + \overline{x}_c - \sum_{u=0}^{w-2}x'_u$$

When verifying the validity of an offshore transaction, the verifier asserts balance by accounting for the collateral and checking the following:

$$\sum_{u=0}^{w-1}C'_u - \sum_{j=0}^{n_T-1}\overline{C}_j - \frac{1}{r}\sum_{j=n_T}^{t-1}\overline{C}_j - \mathsf{Com}(f,0) - \overline{C}_c = 0$$

To assert that the collateral output represents the expected amount from the oracle corresponding to the amount of XHV converted to xUSD in the transaction, the verifier computes the collateral value $\overline{v}_c = O_{\mathtt{off}}(y_T)$ and checks that $\overline{C}_c = \mathsf{Com}(\overline{v}_c, \overline{x}_c)$.

3.2 Onshore transactions

Onshore transactions are modified more significantly. In particular, outputs of both xUSD and XHV asset types may be consumed in such a transaction, and a set of additional distinguished XHV collateral outputs must be generated in the transaction. The value of the generated collateral outputs depends on the result of an oracle query on the value converted in the transaction, and is intended to match the total value of consumed XHV outputs.

Let w_T be the number of consumed xUSD outputs in the transaction, $w_{T'}$ be the number of consumed XHV outputs in the transaction, and set $w = w_T + w_{T'}$.

Let $\{v_u\}_{u=0}^{w-1}$ be the values of consumed outputs in an onshore transaction (in the appropriate units based on asset type). Let $\{\overline{v}_j\}_{j=0}^{n_T-1}$ be the values of `xUSD` outputs generated. Let $\{\overline{v}_j\}_{j=n_T}^{t-1}$ be the values of `XHV` outputs generated.

The transaction is modified to require distinguished `XHV` collateral outputs with values $\{\overline{v}_{c,j}\}_{j=0}^{n_c-1}$, where n_c is the number of such outputs the sender chooses to produce. It is required that the sum

$$\sum_{j=0}^{n_c-1} \overline{v}_{c,j} = O_{\mathsf{on}}\left(\frac{1}{r}\sum_{j=n_T}^{t-1} \overline{v}_j\right)$$

of collateral values match the amount provided by the oracle. When generating these outputs, the sender selects commitment masks $\{\overline{m}_{c,j}\}_{j=0}^{n_c-1}$ uniformly at random and defines the outputs as $\{C_{c,j} = \mathsf{Com}(\overline{v}_{c,j}, \overline{x}_{c,j})\}_{j=0}^{n_c-1}$. The sender includes the mask sum

$$\overline{x} = \sum_{j=0}^{n_c-1} \overline{x}_{c,j}$$

in the transaction data, and includes the collateral outputs when producing an aggregated range proof for the transaction.

When producing masks for the consumed output commitment offsets, the sender accounts for the collateral outputs, but does so in a separated manner that depends on the asset types. Namely, it sets two specific masks in a non-uniform manner:

$$\begin{aligned} x'_{w_T-1} &= \sum_{j=0}^{n_T-1} \overline{x}_j + \frac{1}{r}\sum_{j=n_T}^{t-1} \overline{x}_j - \sum_{u=0}^{w_T-2} x'_u \\ x'_{w-1} &= \sum_{j=0}^{n_c-1} \overline{x}_{c,j} - \sum_{u=w_T}^{w-2} x'_u \end{aligned}$$

The sender modifies the commitment mask sum

$$a_T = \sum_{u=0}^{w_T-1} \overline{x}_j$$

and value sum

$$y_T = \sum_{u=0}^{w_T-1} v_u - \sum_{j=0}^{n_T-1} \overline{v}_j - f$$

to account for the presence of consumed outputs of two types.

When verifying the validity of an onshore transaction, the verifier asserts balance by accounting for consumed outputs of both types and the collateral and checking the following:

$$\sum_{u=0}^{w_T-1} C'_u + \frac{1}{r}\sum_{u=w_T}^{w-1} C'_u - \sum_{j=0}^{n_T-1} \overline{C}_j - \frac{1}{r}\sum_{j=n_T}^{t-1} \overline{C}_j - \mathsf{Com}(f,0) - \frac{1}{r}\sum_{j=0}^{n_c-1} \overline{C}_{c,j} = 0$$

To assert the value of y_T, the verifier checks the following:

$$\sum_{u=0}^{w_T-1} C'_u - \sum_{j=0}^{n_T-1} \overline{C}_j - \mathsf{Com}(f,0) - \mathsf{Com}(0, a_T - \overline{a}_T) - \mathsf{Com}(y_T, 0) = 0$$

To assert that the collateral outputs collectively represent the expected amount from the oracle corresponding to the amount of xUSD converted to XHV in the transaction, the verifier computes the collateral value $\overline{v} = O_{\mathtt{on}}(y_T)$ and checks that

$$\sum_{j=0}^{n_c-1} \overline{C}_{c,j} = \mathsf{Com}(\overline{v}, \overline{x})$$

holds.

4 Implementation

We briefly describe the scoped areas of the implementation.

Note that the provided scope appears not to fully cover all codebase changes relating to the modified protocol. While we have endeavored to make reasonable scope extensions for the purpose of identifying specific behavior and intent, it was not feasible to identify a specific complete scope extension.

4.1 src/ringct/rctSigs.cpp

The logic from `src/ringct/rctSigs.cpp` in `verRctSemanticsSimple2` handles verification of offshore and onshore transactions, using offloaded values for expected collateral and transaction type metadata. It relies heavily on consistency assumptions about consumed and generated asset types and transaction type inference to properly assert transaction balance and perform other value and collateral checks.

The balance logic for offshore and onshore transactions follows the protocol, but does not distinguish generated collateral outputs from non-collateral outputs for balance purposes. Instead, it simply uses the asset type associated to each output to determine its place in the balance computation.

Validation of the claimed amount "burnt" in an offshore or onshore transaction is performed using the claimed mask sums. For offshore transactions, a successful check shows that the claimed burnt amount matches the value contained in generated xUSD type outputs (scaled to XHV units). For onshore transactions, however, a successful check shows that the claimed burnt amount matches the difference in value between consumed and generated XHV outputs (scaled to xUSD units).

The verifier then attempts to reconstruct a generated collateral commitment from the expected collateral value and provided mask sum.

4.2 `src/cryptonote_core/tx_pool.cpp`

Functionality from `src/cryptonote_core/tx_pool.cpp` in `add_tx2` asserts the correctness of lock times for generated collateral outputs in offshore and onshore transactions, which are intended to prevent the spending of `XHV` collateral before an expected height. This functionality relies on transaction-supplied indexing to identify and check a collateral output, and then performs asset type and lock height checks. The same functionality applies a fixed fee of 1.5% of the total amount converted in either an offshore or an onshore transaction, and which is dependent on transaction version. This fee is checked against the transaction's stated offshore fee, and rejected if there is not an exact match; we note that naming conventions in the code imply that the fee should not be too low, and that a fee that is too high will also be rejected.

4.3 `src/wallet/wallet2.cpp`

In `src/wallet/wallet2.cpp`, consumed `XHV` outputs for onshore transactions are selected in `transfer_selected_rct` to match the collateral amount produced in generated collateral outputs. These `XHV` outputs are placed within an ambiguity set of other outputs of the same type, in order to match the approach used for non-collateral outputs.

5 Observations

5.1 Onshore algorithms are not correct

The onshore transaction definition is not correct. While the verifier checks will pass if the sender validly matches the total consumed `XHV` value to that of the generated collateral (in addition to the overall balance requirement), it is not guaranteed that successful checks assert the relationship between these values.

While the verifier's check on y_T does assert that

$$y_T = \sum_{u=0}^{w_T-1} v_u - \sum_{j=0}^{n_T-1} \overline{v}_j - f$$

as expected, it follows from the balance check that

$$y_T = \frac{1}{r}\sum_{j=0}^{n_c-1} \overline{v}_{c,j} - \frac{1}{r}\sum_{u=w_T}^{w-1} v_u + \frac{1}{r}\sum_{j=n_T}^{t-1} \overline{v}_j .$$

Unless the first two summands cancel (which occurs if the sender chooses the values to do so), the verifier will not query $O_{\texttt{on}}$ with the same value as the sender.

The combination of the y_T check and the balance check therefore do not assert that the consumed `XHV` value (the second summand above) separately balances the generated collateral value (the first summand above), which is the intent of the protocol.

One mitigation is to separately check that this expected "sub-balance" relationship holds, and that such a change is implemented accordingly.

5.2 Unnecessary range proof use

In offshore transactions, the collateral output $\overline{C}_c$ is included in a range proof. If the range proving system supports aggregation, this is as part of the transaction's single aggregated range proof. If it does not support aggregation, a new range proof is included in the transaction data.

However, this is not necessary. The verifier checks that $\overline{C}_c = \mathsf{Com}(\overline{v}_c, \overline{x}_c)$ using the mask $\overline{x}_c$ provided by the sender and the value $\overline{v}_c$ returned by its oracle query. Because the commitment scheme is computationally binding, it is not feasible for the combination of sender and oracle (even if colluding) to produce another valid opening for $\overline{C}_c$. This means that the verifier need only assert that $\overline{v}_c$ is within the valid range specified by the protocol (presumably, the same range as enforced by the range proving system), and the inclusion of this commitment in a range proof is unnecessary.

It is important to note, however, that this analysis does not apply to the case of onshore transactions. In that case, the sender may have produced multiple collateral commitments. The verifier's check that

$$\sum_{j=0}^{n_c-1} \overline{C}_{c,j} = \mathsf{Com}(\overline{v}, \overline{x})$$

does not assert that any of the commitment summands have a sender-provided opening corresponding to a value within the valid range. It is required that each commitment summand be included in a range proof to make this assertion.

5.3 Onshore mask computation can be simplified

When selecting masks for consumed outputs in an onshore transaction, the sender samples all masks uniformly at random except for one mask corresponding to an `xUSD` commitment and another corresponding to an `XHV` commitment. This manner of selection is consistent with the verifier's balance check, which requires a particular weighted sum of consumed and generated output commitments be zero. However, the use of two non-random masks is unnecessary.

The sender can sample all consumed output masks uniformly at random except for only one, and account for the necessary weighting. In particular, it can sample the masks $\{x'_u\}_{u=0}^{w_T-2}$ and $\{x'_u\}_{u=w_T}^{w-1}$ uniformly at random, and then set

$$x'_{w_T-1} = \sum_{j=0}^{n_T-1} \overline{x}_j + \frac{1}{r}\sum_{j=n_T}^{t-1} \overline{x}_j + \frac{1}{r}\sum_{j=0}^{n_c-1} \overline{x}_{c,j} - \sum_{u=0}^{w_T-2} x'_u - \frac{1}{r}\sum_{u=w_T}^{w-1} x'_u$$

as the remaining mask.

Aside from possibly simplifying mask selection (depending on implementation details), this approach has a second benefit. As noted in previous analysis, the nature of the verifier's balance check is such that the masks used in a transaction's commitments have a specific weighted relationship necessary for the check to pass. In the original Haven Protocol design noted in that analysis, this means that knowledge of all but one commitment mask (between consumed output commitment offsets and generated output commitments) trivially leaks the remaining mask, which corresponds to an information leak. In the updated design, mask selection is such that it is possible to leak a commitment mask given a smaller number of other masks (depending on the type distribution).

We note that the previous analysis recommended a safer approach to mask selection and balance assertion that removes this problem entirely. In that recommendation, the sender chooses all masks uniformly at random for consumed output commitment offsets and generated output commitments. Then, the balance assertion is replaced with a representation proof of the same weighted commitment sum as used in the original design. The same approach may be used here.

5.4 Collateral masks can be hidden or removed

In both onshore and offshore transactions, information about collateral masks is revealed.

In the case of an offshore transaction, the mask $\overline{x}_c$ is included in the transaction data, and the verifier reconstructs the collateral output commitment to check its value. In the case of an onshore transaction, the mask sum

$$\overline{x} = \sum_{j=0}^{n_c-1} \overline{x}_{c,j}$$

is included in the transaction data, and the verifier reconstructs the collateral output commitment sum to check the total value.

In both cases, revealing mask information directly is not necessary. For offshore transactions, the mask can be set to any public value (like zero) and excluded from the transaction data altogether; the verifier then reconstructs the commitment from this public value. There is no security gained from using a random mask that is publicly revealed.

For onshore transactions, the situation is more complex. In the case where $n_c = 1$, this is identical to the offshore case. In the case where $n_c > 1$, revealing the sum of all masks leaks information; when coupled with information about other masks, this may reveal individual collateral output commitment masks.

A comprehensive solution that applies to both cases is to continue selecting these masks uniformly at random, but to replace the mask sum with a representation proof that the total value is correctly represented by the commitment sum. That is, the sender produces the proof

$$\Pi_{\text{dl}} = \mathsf{DLProve}\left(pp_{\text{dl}}, \mathsf{Com}(0,1), \sum_{j=0}^{n_c-1} \overline{C}_{c,j} - \mathsf{Com}(O_{\mathsf{on}}(y_T), 0); \sum_{j=0}^{n_c-1} \overline{x}_{c,j}\right)$$

and includes it in the transaction data in place of the mask sum $\overline{x}$. Then, the verifier checks that

$$\mathsf{DLVerify}\left(pp_{\text{dl}}, \sum_{j=0}^{n_c-1} \overline{C}_{c,j} - \mathsf{Com}(O_{\mathsf{on}}(y_T), 0), \Pi_{\text{dl}}\right)$$

succeeds instead of reconstructing the expected collateral output commitment sum from the mask sum. This method asserts that the total value is represented by the given collateral output commitments without leaking the corresponding mask sum to the verifier.

5.5 Inconsistent handling of collateral outputs

The protocol dictates that an offshore transaction generate a single collateral output, and that an onshore transaction generate one or more collateral outputs.

In `tx_pool.cpp`, the checks for lock time correctness rely on transaction-supplied indexing to identify and check a single collateral output, and then perform asset type and lock height checks. If there is a mismatch in the indexing, it will not be detected. Further, these validity checks will not properly perform their assertions in the case of an onshore transaction with multiple generated collateral outputs.

In `rctSigs.cpp`, the sender populates the collateral mask sum based on transaction type. At that point, it does not assert that an offshore transaction contains only a single generated collateral output, but appears to populate the mask sum with the last such indexed output. For an onshore transaction, it only populates the mask sum if a generated collateral output exists of the expected value; while this is technically valid according to the protocol description, it does not reflect the full generality of the description.

In `rctSigs.cpp`, the verifier checks that a generated collateral output is valid using the expected collateral value and provided mask sum. However, it does so only using a single generated collateral output corresponding to a provided index. This will fail if an onshore transaction uses multiple collateral outputs.

5.6 Inconsistent use of fees

As before, the protocol specifies only a single fee provided in the non-converted type of any transaction, while the implementation includes two fees of this type: a “standard” fee and an “offshore” fee that is used for both offshore and onshore transactions. These fees are combined for the purpose of balance computation in a safe way, but determination of valid fee values is out of scope and offloaded elsewhere in the protocol.

5.7 Repurposing of transaction flows

The implementation generally reuses or repurposes existing transaction functionality or data structures where possible, presumably to reduce duplication of code and maintain a smaller and more unified transaction flow footprint. While this can be useful when done with sufficient care and generality, it introduces risk; the semantics of offshore and onshore transactions, and the differences in the resulting verification logic, are related but not identical.

As a result, the areas of code in scope, as with other previous transaction types, rely on often subtle relationships between output types and transaction fields to infer metadata about a transaction for the purpose of determining the correct logic to use. This approach makes it challenging to assert the absence of edge cases that could result in the incorrect logic being applied. We strongly recommend the addition of comprehensive tests that exercise possible failure modes, whether intentional or not, that could arise from transactions with unexpected or inconsistent semantics.