In the Multi-Asset Shielded Pool, the limited additional fungibility provided by the Convert circuit allows for fungibility among tokens with distinct metadata, such as time value. For example, different “vintages” of the same token can be distinguished inside the pool by epoch shielded. By re-introducing fungibility to these now distinct vintages, a (public) conversion rate can apply to tokens based on this metadata and therefore a different rate or even a different primitive token type applies to the conversion as a function of the metadata. The most direct application of this is “integral evaluation”: applying some function based on the quantity of the token multiplied by the amount of time a token has existed in its present state, despite the fact that both the quantity and time are private values.

The Convert circuit can work well when this function changes frequently; for example, conversion rates can be computed outside of the circuit (although it can still be computationally expensive to do so) and when only the latest conversion rate applies. For example, if tokens can only be Converted to the latest vintage, this is better both for privacy and efficiency.

A more substantial challenge occurs when attempting to do this in a shielded pool that doesn’t apply such metadata to tokens, or that applied some other metadata. For example, an existing shielded pool probably wasn’t designed to allow this integral evaluation. Alternatively, it might be desirable to do this integral evaluation over an arbitrary range of time and not only ending at the present time. Unfortunately, over n epochs there would be O(n^2) possible integration domains to compute conversion rates, which is far too inefficient.

So let’s assume a slightly different context than the Convert circuit does:

1. Tokens don’t have metadata attached (of any kind) - only quantity and type
2. The function to compute is a simple function of the quantity * time, for example, a constant * quantity * time
3. For each note commitment (or nullifier) it is publicly known which epoch that note commitment (or nullifier) was revealed

Then we can define an integration circuit which takes as public input:

- A merkle root rt_cm of a tree with leaves (note commitment, i) where i is the epoch that note commitment was revealed
- A merkle root rt_nf of a tree with leaves (nullifier, j) where i is the epoch that nullifier was revealed
- A value commitment to the output of the function
- An authorization signature
- An integration nullifier nf_I

and checks the following:

- there exists a merkle path in rt_cm to a note commitment and epoch (cm, i)
- there exists a merkle path in rt_nf to a nullifier and epoch (nf, j)
- nf is the correct nullifier of cm
- the authorization signature was created using the spending key in cm
- the value commitment opens to c * v *(j - i) where c is a constant and v is the value (quantity) in cm
- nf_I is correctly derived from cm and nf in a proper way

Then the authorization signature and nf_I are checked outside of the circuit for validation and uniqueness, respectively.

The value commitment can be used either as public input to another circuit, or possibly used to balance a shielded pool transaction, if the value represents a claim of some tokens.

A note on proof systems and curves: in addition to the problems described above, the proof systems and curves necessary for this present new problems, especially when combining different pools. Considering several concrete examples:

1. BLS12-381/JubJub based pools (Sapling, MASP). Since the note commitments are over the JubJub curve, an integration circuit must be implemented in the JubJub base field. If the proof system used is groth16 over BLS12-381, then the integration circuit requires another trusted setup.
2. Pallas/Vesta based pools (Orchard, Taiga). An integration circuit wouldn’t need a new trusted setup, but also would not be immediately compatible with the value commitments output by a JubJub integration circuit (or vice versa)

We can try to mitigate this a little bit. The value commitment could be based on blake2s (or other hash) and an additional circuit on a different curve can convert the value commitment to that curve.

Another mitigation would be to use Halo2 as the proof system for a JubJub based integration circuit. BLS12-381 could be used with Halo2 at the expense of being inefficient. A new elliptic curve could be found with prime order equal to the order

of JubJub's base field. This should be possible in principle but so far no such curve is known.