At the application layer, concepts like ownership of resources are important for many different reasons. Ownership, in the sense that a specific identity can permit or prohibit the consumption of a resource it owns

, does not exist at the resource machine level, so we need to implement it on top of that.

Resource consumption in the resource machine

To consume a resource at the lowest level, two things have to happen:

- A nullifier has to be created.
- The resource logic has to be satisfied.

Satisfaction of the resource logic depends on the contents of a TX containing the resource. Computing the nullifier, requires knowledge of the plaintext resource. These two requirements are not tied to each other by default, s.t. anyone who knows the plaintext of a resource could authorize consumption within a TX satisfying the resource logic.

Implementing consumption authorization in resource logics

To tie the two requirements together and close the gap we could implement ownership tied to an identity in a resource logics as follows:

- Pick an encoding convention for placing an external identity (or a commitment to one) in the value field of a resource to determine the current owner of the resource.
- Add a check to the logic that checks wether this identity signed over a transaction involving the resource.
- Add a check to the logic that the information required to compute the nullifier is verifiably encrypted to the external identity of the new owner of a resource created after consumption of the current one.

Thanks to

@vveiln for explaining how consumption works as a primitive and@cwgoes for discussion on the resource logic implementation.