This discussion began on slack.

Summary to kick off discourse discussion

Prior to this discussion (pasted below), I thought that all nullifiers would be the hash of the corresponding note commitment and the user's private key. This discussion made it clear that that is not necessarily the case. An app circuit is responsible for defining nullifier structure

which may or may not include information that ties a nullifier to a user.

There are some concerns that this will create a tough developer experience in which it is easy for developers to create nasty bugs. Hopefully some standard Noir libraries can minimize such issues.

Slack conversation

Lasse

:

If the nullifier is dependent on the users private key, cant I get into issues where two users can consume the same data because they got different private keys. It should only depend on the data in the commitment or derivable

Mike

:

I think it's the app circuit's job to verify this kind of thing. I'm pushing for us to enable custom nullifiers, where the app circuit determines what the definition of a nullifier is. In that paradigm, the kernel circuit should just blindly insert the nullifier into the nullifier tree.

David

:

So two users could theoretically submit the same nullifier, but of course the rollup circuits would reject the second.

But in this paradigm you mention Mike, app-circuits are fully responsible for validating nullifiers (or just not validating them at all), but then I imagine they will still be siloed by contract address.

When I say validating, I mean "ensuring that nullifiers are constructed properly". I know that membership checks are handled in rollup circuits

Mike:

So two users could theoretically submit the same nullifier, but of course the rollup circuits would reject the second.

Yep. Like, in Aztec Connect, we had nullifiers for aliases. This was just hash(alias). It meant if two people tried to register the same alias, only the first person would succeed.

app-circuits are fully responsible for validating nullifiers (or just not validating them at all), but then I image they will still be siloed by contract address.

Yeah app circuits can define nullifiers to be any field element, derived in any way which makes sense for the app. But yeah, they'll always get siloed by the kernel circuit. Otherwise I could build a (malicious) app, which can emit as a nullifier whatever value I input as an argument. Then I could just ruin lives, by watching the tx pool, and front-running every single nullifier.

David:

And it would be up to the app to set the rules for who can nullify a certain commitment, correct? Or would only the "owner" be able to nullify?

Lasse:

There might not be an "owner" dependent on the commitment definition by the contract

David:

Okay I see

Lasse:

Owner could be anyone that knows the content.

But seems like we will be in for a collision hell if people write some bad code.

David:

Yeah

Lasse:

Might be similar to people writing bad code and getting hacked. But less obvious in some of these maybe.

There is quite a few footguns with privacy.

Mike:

Yeah if an app developer tried to design their own note, they could cause nullifier collisions if they don’t know what they’re doing.

Once Noir generics are working, we’ll be able to write some very sophisticated ‘standard library’ versions of notes and nullifiers, for ‘normal’ devs to safely do a lot of cool things.