The oddities

When reviewing specs V2, I noticed that the Identity Machine felt very peculiar to me:

specs.anoma.net

Identity Management Engine - Anoma Specification

The Anoma Specification is a collection of documents that describe the architecture, design, and implementation of the Anoma Network.

Namely that everything is an engine!

1. reads-for
2. signs-for
3. naming
4. commitment
5. …

Being an engine means a few things:

1. It can not something the system can natively talk about (Nothing in our execution language implies being able to create or run Engines)
2. I.E. we can't create these ideas in anockma, they are speical forms

of the system

1. I.E. we can't create these ideas in anockma, they are speical forms

of the system

1. It is something additional that must interact with user flow in some way
2. I.E. storage offers a storage service that the client environment and distributed node state can use.
3. I.E. storage offers a storage service that the client environment and distributed node state can use.

Since the condition of 1 requires no thought, let us tackle what 2.

means for some of the identities engines.

I want to use signs-for

and the identity-management

engines as my examples for this post.

For the signs-for

engine, we can conceptualize the use of it within the anoma system, as providing a primitive function in anockma

.

Namely as part of the actions that anockma execution returns can be a signs-for

message that gets propagated to the signs-for

engine that is a special genserver in the implementation.

Likewise if we want to connect or generate keys we can expose the identity management

engine to execution by offering anockma

actions that it can return that signal to the identity management

engine our desires.

What this is stating is that we aren't even attempting to handle these kinds of events "in Anoma", rather we lay out bare that these concepts are not easily conceivable with the primitives we have in Anoma and would rather hard code in this

behaviour.

A directionality correct flow

Now let us imagine a design for the identity machine that tries to utilize Anoma as a system.

Some of these “engines” offer an interesting challenge. Namely the identity management

“engine” will require some care.

Before we can begin to describe these engines, we must first lay bare how inside the belly of Anoma and describe how Anoma can store data

Cryptography in system

The first kind of cryptographic keys I wish to talk about are ones that live inside the system.

Thankfully, anockma

is just nock

and thus we can use it to generate keys such as ed25519

, thus key generation is well within the power of the system.

Now what’s nice about anoma, is that we have two kinds of storage:

1. distributed gossiped storage (on “node”)
2. local non gossiped storage (on “client”)

In this system we can store these private keys in the non gossiped storage.

We can do it by hash (via blob storage), or by path. This is achievable because we can scry (e.g. /blob/hash

for blob storage). With this, we can store our key somewhere where we know.

Thus we can imagine simply generating out this key with some anockma

and then storing in a designated path. From here we can have an anockma

function that can list you all known keys.

Now that we have storage out of the way, let us talk about the signs-for

“engine”.

We can store signs-for

data in distributed gossip storage, as everyone would need to know this fact if they wish to attest to that Alice

signs for Bob

.

Now, since signs-for

sounds like a set (maybe Mallory

can also sign for Bob

), we can’t store this easily using distirbuted blob storage, instead I suggest we use our append-only storage to save it on the path

Bob's public key/signs-for

, which the user can then sign for using something like this

scry distributed/Bub's public key/signs-for

.

Thus when we have some resource logic, we can invoke the signs-for?

predicate in anockma

that appropriately scries and checks against the list (ZK implementation note, you can probably pass anything you wish to scry before calling prove all fine as this data is available to scry then pass into the ZK circuit, meaning this is all possible).

Great, we've now taken the signs-for

and part of the identity-management

engine in Anoma itself!

Cryptography out of system

Now we have to face the reality that some data may exist outside of the Anoma system! This can happen when we want to access resources on the underlying operating system that we are apart of.

This may commonly happen to access files on system, or maybe to load some code into the system.

For the purposes of this post, this most happens with the identity-management

"engine", namely we may have some external keys someone generated on say "ledger".

In this case, this requires a bit of special care, but only enough special care to internalize outside resources.

For a file, this can be done easily by interpreting the scry path as an external file system path, getting back the contents in which we can then add to blob storage or program over. Likewise for a port or a connected device that signs signatures it likely will require some extra external care.

However once this care is given, the type can be unified into the model of how we view signing and thus the identity-management

"engine" can be conceptualized as in part special in how outside resources connect (but this is not an engine) and then mostly within Anoma as an interface for how "identities" work.

(This is nothing new, and has been done a million times over with ports in basic image systems…)

However, these extensions need to be developed and thought of, I believe time spent thinking about how the Anoma system (not software on UNIX/Windows) can utilize the underlying OS and develop the Anoma system will be vital to Anoma as a system working, as making every component special will delay this development as we fail to get comfortable with Anoma itself.

I think in the future I'd conceptualize all of these in AL as giving me programable resources that will have the same method interface whether it's stored in Anoma proper or is an outside resource. This distinction has not harmed how one uses proxy objects in smalltalk or external resources in language as the model is strong enough to make external data look internal to the user, which is the property we want.

Conclusion

The current spec proposes that identities are special and non derivable, and further has not contemplated on how it will be used in the system. I give an example of how we might make these special components apart of the system while still being special. However, the main thrust of this post outlines that "identities" aren't engines at all and can be conceptualized within Anoma with development being placed in places that help complete the system.

We all need to stop worrying and love Anoma