While I was working on writing upthis discussion in this PR, I realized that we still don't have a proper general proof type or multiformat, and I think we need one. In particular, I think that we want a proof type which can encompass "heterogeneous trust" / "heterogeneous assumption" proofs, including, for example:

- A trivial

proof where computation is simply replicated, and the data used as input to the computation is packaged in the proof. It has no extra security assumptions but is not succinct.

- A trusted delegation

proof where computation is delegated to a known, trusted party whose work is not checked. This scheme is succinct but requires a trusted party assumption (which could be generalised to a threshold quorum in the obvious way). Note that since the computation is still verifiable, a signer of (a, b, predicate)

where predicate a b = 0

could be held accountable by anyone else who later checked the predicate.

- A succinct proof-of-knowledge

proof where the result of computation is attested to with a cryptographic proof (of the sort commonly instantiated by modern-day SNARKs & STARKs. Succinct proof-of-knowledge schemes provide succinctness as well as verifiability subject to the scheme-specific cryptographic assumptions. They may also possibly be zero-knowledge

, in which the verifier learns nothing other than predicate a b = 1

(in this case, and in others, a

and b

will often be "hidden" with hash functions, such that the verifier knows only hash a

and hash b

but the substance of the relation obtains over the preimages).

This implies some slightly different requirements than would a simple "proof multiformat". Let's assume that the verifier - when verifying the proof - specifies which assumptions they're willing to make, which might include cryptographic assumptions (as variously required by various SNARK/STARK schemes), trust assumptions (as required by a trusted delegation proof as described above), etc. These different trust assumptions (made by the verifier) should not, in general, entail different multiformat types - the set of possible trust assumptions is infinite, so that wouldn't work - and we want e.g. trusted delegation proofs made by different parties to be one multiformat type, not multiple. I think a suitable multiformat might look more like the following:

- We have some type of assumptions Assumption
- prove_n

generates the proof of type Proof_n

- compatible_n

checks if a proof of type Proof_n

is compatible with a selected set of assumptions (returning true/false)

- verify_n

verifies a proof of type Proof_n

Question: should compatible_n

and verify_n

simply be combined?

Now we've shifted the problem to what exactly this "Assumption" type is. Without loss of generality, I think we can characterize assumptions as beliefs about logical implications (a \implies b

for some a

and some b

), where, for example:

- a

could be that a particular trusted party has signed over a statement, and b

could be that the statement is true

- a

could be that the algebraic group model holds, and b

could be that a particular proof system is sound

- a

could be nothing, and b

could be that c = c

(some identity case)

In an ideal world, we could characterize all of these assumptions exactly (e.g. as mathematical statements) - however, that will not be feasible in the short term (precisely expressing cryptographic assumptions will require a sophisticated specification language), so we probably need a simple easily extensible sum type that can evolve along with the Proof

multiformat (to add new types of assumptions). The encoding here needs to allow for additions to the sum type without breaking backwards compatibility (which may be a property we often want in general).

I think that this should be sufficient - in principle - to allow for something like smart multievaluation which can e.g. use trusted third party delegation automatically.

Thoughts? @ray @Jamie @ArtemG @AHart @vveiln @degregat