we should have shippable signsfor proofs. the basic problem is that A would like to prove to B that X said something, but what it in fact has is a commitment from Y to that thing, where Y signs for X. there are two problems with making this implicit:

1. there is no global signsfor database; every node will have its own signsfor database (which may change over time); a node cannot remember every piece of signsfor evidence it sees, because that would be a trivial dos attack. so it may be that A has evidence showing that Y signs for X, but B does not
2. even assuming the individual pieces of evidence are shared, signs-for is transitive, and computing the transitive relation is equivalent to graph reachability, which is O(hard). the onus therefore should not be on B to compute it; if A wants to convince B of something, then it should be A's responsibility to do the hard work

therefore: A should be able to find and share the specific path through its signsfor db that shows what it wants to show, producing a proof that B can straightforwardly verify in linear time

compositional identities complicate matters slightly. it can probably be taken for granted that X signs for X \lor

Y. but things get more interesting when you have to mix cryptography with logic; what about, say, showing that P \land

Q signs for X \land

Y \land

Z from the fact that P signs for X \land

Y, and Q signs for Y \land

Z? i don't think it's important to be able to infer that sort of relationship automatically right now (if ever—i'm not sure how to do it and i'm not sure what the point would be), but we should make sure the proof format can handle everything

probably something like the following? define some primitive inference rules, and the proof is a set of cryptographic signs-for proofs plus a sequence of directed rewrites (each justified by an inference rule and the cryptographic proofs) which takes you from the 'source' id to the 'target'. (maybe we should do cnf/dnf implicitly, maybe that simplifies the proof encoding or something? no because renormalising after each step is probably O(bad))