Crossfunctionality

A business developer's guide to Semantics, and an engineer's guide to trust frameworksSuggest Edits

Having worked for years in various corners of the decentralized identity sphere, I have developed a pet peeve for "technosolutionism": the notion that a sufficiently innovative technology can solve a social problem, or a business problem, without complex and nuanced changes to social and business practice. No less unnerving, perhaps, are the brute-force capitalists that chauvinistically proclaim that with enough momentum, traction, and capital, any Betamax can be relegated to the dustbin of history by a vastly inferior VHS. Both views are based on facts, and yet both are also dangerously blindered. Real progress cannot be attributed to TED-Talkers, to technologies that magically conquer markets, or to market-makers that appear publicly as lone snake-charmers working miracles. Real progress is made by rich, cross-disciplinary teams and heterogeneous coalitions coming together to attack hard problems from every angle at once.

Verite is one such coalition, in that it takes a truly cross-disciplinary approach to industry-wide problems. It is rooted in a multi-stakeholder design process, deeply informed by a complex regulatory strategy and the legal needs of various industries. It proposes an end-to-end interoperability stack for portable data, optimized for end-user control and data minimization. It supports centrally-planned, "cathedral" architectures, but equally supports more self-custodial and decentralized "bazaar" approaches, with various extension points and governance possibilities baked into the core building blocks. It is a scalable technology designed to be co-governed by the ecosystem that grows around it, with a calculated resistance to platform plays and lock-in strategies that would usher in platform economics.

But how does it work? That requires a bit of cross-discipinary understanding, so bear with me while I explain things cross-wise.

Semantics for the non-technical business developer

Cryptography makes getting data from one context to another fairly easy-- insofar as you can look up someone's current, valid public key, you can bejust as confident of any data they sign with the corresponding private key. If that data includes a timestamp for the signature, an expiration date, and a way to check it hasn't been revoked, you're dealing with some pretty trustworthy data. This is theeasy part of data portability.

The harder part is what software engineers colloquially call "Semantics": the fine art of knowing what the heck any of those data actually meant in their original context. The data passed around in signed, portable form is only as useful and trustworthy as you arecertain of what each "true", "false", "null", "", "3/5" and "pending" meant to the signer at the time of signing.

If that sounds esoteric, it might help to put on your business developer hat and think through a concrete example: what exactly might "AML_check":"true" mean? It is easy to imagine situations where not all AML checks are created equal, or doors which only more rigorous AML checks should open, perhaps at different price-points or with different liability/insurance requirements. Who operated the check, and what exact definition they checked against (on what data points, against what jurisdiction's requirements, etc) are questions well beyond true|false . At this point, it should be quite clear that "semantics" is not just a required course in Computer Engineering programs: semantics is where the rubber of software design meets the road of real-world law and business.

Semantics from the Verifer's POV

If semantics is how engineers scale up "reliance" (i.e., how one company can safely rely on other company's data), we need to walk through the semantics not from the issuer's POV, but from the Verifier's. They are, after all, betting big on Other People's Properties.

So let's take a little tour of the properties encoded in the "schemas" (sometimes called "schemata" or "credential definitions") that Verite authors and manages over time. Each schema is something like a recipe for data points, what engineers call the "data shape" of each credential or token. Take, for example, thev0.0.1 schema for a KYCAMLAttestation credential: what you see, in machine-readable script, is a list of the mandatory and optional properties that each credential needs to have, and the "type" of each (string, integer, etc.). Let's walk through some specific properties to explain the options.

ApprovalDate

All verifiable credentials contain an issuance date, but crucially, this is the time of signature, or to put it another way, the time of exporting the data and putting it under the direct control of the user; the underlying decision about considering a user "KYC'd" might have happened minutes, or days, or months before. For this reason, a distinct date property is included, which is the date of thedecision ; it is usually more accurate to require freshness of the KYC decision than the data artifact.

Process

One of these properties, process , is not really a simple, singular property like the others. It is a string, but that string actually contains a link to a second schema; in the example above, taken from the tutorials and examples of the current sample implementation, all the KYCAML credentials point tothis process definition . As you can see, this second schema outlines at a high level the real-world process used to arrive at the first schema's data points (in machine-readable format, which can include links to additional machine-readable documents and/or to human-readable contracts or legal terms). Or, to put it another way, this defined what the issuer meant by "AML_check" in more or less detail; more rigorous and production-ready process definitions could include links to which authorities or registries were checked, maximum time between checks and issuance/decision date, etc.

There is a 1-to-many relationship between credentials and processes; many different processes can be used to arrive at the same bundle of distinct claims, and issuers operating in multiple jurisdictions often juggle many different processes for different customer bases. Not all processes should be assumed to be equivalent; in fact, the whole point of schematizing them is to allow precision in defining their difference and choosing which to accept in a given situation. The closer this kind of system gets to production, the more likely this process of refinement will involve getting into the weeds and talking to lawyers!

We are assuming that over time, more and more processes will be added to the system, as more and more issuers come on line and want to define precisely what conclusions and liabilities they are comfortable with verifiers extracting from their credentials. That said, as competition grows, many will simply conform to existing credentials to simplify proof of "equivalence". I.e., even if their credentials are checked to a more complex or detailed process, new issuers may choose to issue credentials only containing what their competitors' credentials already cover, to simplify the due diligence of those looking to accept both interchangeably.

Issuer

The mention of equivalence between competitors lays bare something often implicit in the structure of Verite and in the design of verifiable credentials more generally: even if the claims made by two issuers are equivalent and interoperable. Identical credentials issued by different authorities are never truly interchangeable, because the identity of those authorities matter, as does reputation, history, marketing, jurisdiction, and licensing. Every verifier decideswhose credentials it wants to accept, according to its own due diligence process and risk-management decisions; "signing on to Verite" does not forfeit that right at all.

On the contrary, over time we hope Verite will build and iterateautomated mechanisms that reduce the complexity and work involved in manually onboarding and vetting issuers. Once there are enough issuers for a given credential type, anchored to process definitions and pegged to requirements for different use-cases and/or jurisdictions, manually allow-listing issuers will not be a scalable approach. When that day comes, we hope Verite can house some form of collective-diligence mechanism for deciding which issuers to trust for which credentials, something like a recommendation network or a vetting authority. This might use token-curated registries or other on-chain artefacts, or their auditing might be off-chain, depending on the use-case and parties involved.

Composability: Extension-points and Decision-points

Having looked at the schema for individual credentials, we've discussed how this kind of data can safely scale and be relied upon. But with the mention of registries or other mechanisms to scale up equivalence (between issuers), we start to see the big picture of scalable and composablereliance . Companies need decision trees of whom to accept as a customer or how to onboard customers based on portable data they bring from elsewhere. Automating that requires lots of flexibility, composibility, and fine print. Engineering for real-world trust across contexts is hard work. (And thanks for making it this far, I know it's a long tour!)

Fundamentally, Verite is a system of composable data formats that let people encode complex reliance and liability relationships in simple schemas, and thus a forward-commitment to interoperability. Each data format can be thought of as one link in a "reliance chain", regardless of whether the linkages are between divisions or subsidiaries of one company or across company lines.

Putting aside a little the question of how many different entities are present in the chain, you could also think of each linkage between different credentials and processes and issuers as an "interface", allowing the end-to-end system to be highly composible, mixing and matching credentials and data points as needed.

A third way of conceptualizing the whole is to take an inventory of all the "decision points":

- Each issuer chooses which credentials to issue,
- Each credential issued refers specifically to the process definitions it used to arrive at those claims,
- Each issuer can choose which kinds of wallets to issue these credentials into, based on technical and/or business criteria
- Each wallet-holder can inspect these credentials and, with a little effort, understand them; nothing in them is opaque or proprietary
- Each verifier can choose which issuers, which credentials, and which process definitions are adequate for a given use-case* If that much choice is overwhelming, verifiers can choose to trust "bundles" or sets of credentials, process, and issuers recommended by Verite (or other authorities)

Add to this that anyone can join and propose additional credential schemas, process schemas, and digital or real-world trust mechanisms. The superpower

Conclusion: Anti-fragile Identity Solutions

Over time, we are expecting more and more implementers to show up, allowing a competitive ecosystem to emerge for multiple roles in this system. No one should fear this kind of competition, though, at least, no one interested in decentralization. If anything, building in equivalence, semantic flexibility, and shared governance into an open-ended identity system is just common sense, given how uncertain the future of that system's requirements looks today.

To name just one unpredictable factor, regulations on international capital flows, digital asset liabilities, and reporting requirements are all coming fast worldwide. These will fundamentally evolve and continue to iterate the groundrules for what level of identity assurance, auditability, and sharing is required for various kinds of economic activity; more crucially for Verite, it will also iterate how directly or indirectly these requirements can be met, by what kinds of reliance.

The more reliance is allowed, the more competition will arise in the cottage industry of distributed identity-checking and verification; to stabilize and regularize this reliance, full-blown "trust frameworks" (multilateral reliance agreements) could emergy quite quickly as they have in other identity markets, particularly if the regulators of major markets demand both auditability and healthy competition outside of proprietary platforms.

To be ready for this development, we recommend everyone trying to pin identities onto financial actors think hard about how flexible and composable their trust system is. If it's not regulation-ready and winter-proof, if it won't survive a sudden redrawing of allegiances and a reorientation around new networks and flows, is it really fit-for-purpose? Identity problem necessitate lawyer-friendly engineering... but also open-ended, antifragile engineering, if it's going to survive the roller coaster of crypto. Updated5 months ago