Overview

Suggest Edits

Landscape

"Knowing your customers" in the traditional banking sense is a tall order, and a one-size-fits-all regulatory approach that can be a huge burden on crypto businesses, that often need 1/10 of the certainty afforded by KYC'ing all their customers or counterparties and are loathe to assume huge costs and liabilities just to do it. Add to that rising regulatory risks and liabilities... and you've got every reason to let a trusted intermediary KYC your customers for you. Verite lets you get just the bare minimum you need from a more dedicated identity verifier, with a verifiable, trustless passing of basic facts. If you ever need to prove exactly when and how you approved a blockchain address or a wallet to interact with you... you've got timestamped, tamperproof, receipts. As ironclad as on-chain data!

Some key questions to ask yourself:

1. What use-cases do you want to start with? Overview of use-cases
2. What is the bare minimum you need to know to allowlist/grant access to an address or a wallet? Data minimization
3. What kind of wallets is it strategic for you to support? Wallet Overview
4. What does "reliance" mean, given the above? Liability and Auditing Considerations
5. Is the issuance/wallet-interaction something you want to "build or buy"? Architectural options

Use-Cases

In the short-term, the following use-cases are going live in at least one product offering in 2022:

1. VC-based gating of KYB status and domicile (US y/n) for a "company wallet" (i.e., Compliant and Auditable Institutional DeFi)
2. VC-based gating of Investor Accreditation for a "company wallet" (US-domiciled wallets)

The following are in research and design phase, being co-developed by participants and adopters:

1. KYC'd individual wallet (custodial and self-custodial)
2. Non-US KYB status (including interoperability with GLEIF and FincID credentialing)
3. FATF-compliant reporting for custodial-to-custodial transactions (incl custodial-to-noncustodial transactions)
4. Currency controls/FX reporting
5. Verifiable credit-assessment and forensics-sourcing

Data Minimization

In many use-cases, the least you can get away with knowing about an address is its:

1. functional jurisdiction (tax residence, domicile, etc is often used as a proxy for this)
2. liability (personal wallet or "company wallet", i.e. funds and liability belonging to a legal person)
3. control model (including identity assurance, possibility of recovery, hardware/software security, and other wallet capabilities)

None of these are PII, and if sufficiently coarse-grained, none of these compromise on the anonymity central to the ethos of cryptocurrency and web3. Verite exists to make these kinds of facts verifiable, yet still under direct user control with a minimum of issuer involvement or surveillance risk.

If you need anything finer-grained than the above, you inch towards the risk of deanonymizing the wallet/user, leaking personal information to be snatched up by prying scammers or spammers, etc. To get into that more dangerous territory, you will have to proceed to request further information from the wallet and/or issuer; this can entail a more niche zero-knowledge protocol or circuit/setup, or a direct channel to the issuer and authorization from the user to inquire about sensitive data they retain.

Liability and Auditing Considerations

The best way to think through reliance from a legal and compliance point of view is this: what are you inferring or taken as evidenced, once you've verified a valid (non-revoked, non-suspended) Verite VC from a trusted issuer? You are responsible for all that you inferred or assumed, and the issuer is responsible for only what is in the credential. For this reason, it is highly recommended that you log the verification interactions carefully, retaining at least the issuer's identifier and whatever the issuer might need to produce the VC itself, for forensic purposes.

A note on "Uptime": if you are consuming credentials that may need to be revoked quickly, such as credentials reflecting the current status of real-time monitoring of data sources like OFAC and PEP lists, you may need to check the "status" of a

credential far more frequently than you verify the credential itself. Luckily, this status can be expressed as a web URL, an on-chain registry, and/or as a value queried via on-chain oracle; unlike the credential itself, no fancy cryptography is needed for these more frequent checks.

Wallet Overview

Take a minute to ask yourself some difficult strategic questions:

- Depending on your business model, you may be more interested in supporting "identity wallets" (applications for signing contracts, handling sensitive identity documents, providing verifiable consent, etc) or in supporting "cryptocurrency wallets" (that authorize transactions on cryptocurrency blockchains and "web3" applications).
- You might be interested in supporting only cryptocurrency wallets with full "identity wallet" functionality, or interested in separating the two concerns in two distinct pieces of software. (Our sample implementation may provide a useful starting point for this latter approach! A white-label browser-extension for identity functionality to complement a cryptocurrency wallet is coming soon).
- Retail wallets tend to have long, slow upgrade cycles and governance processes. Conversely, many companies contract out to wallet firms to provide highly-customized "provisioned wallets" to their employees for managing company funds. As Verite capabilities are standardized and rolled out as common APIs, these may be a better match for "testing the waters"
- Depending on which exact credentials you issue and your risk tolerance, you might have different requirements for identity-assurance, sybil-resistance/uniqueness, deduplication, or liveness/biometric binding. I.e., if your use-case requires you to be certain that the authorized employee is authorizing each transaction of a company wallet, you may want to limit your support to wallets with built-in per-transaction or per-session biometrics, etc.

Architectural options

At present, the two main options to consider are whether you want to attest to the controller of anblockchain address or to the controller of a specificwallet , which may control multiple addresses in addition to a DID (wallet identifier). For more information, read theidentifier scheme considerations and compare theaddress-bound credential exchange flow and thewallet-bound credential exchange flow .

For address-bound flows, please seeWallet Connect request presentation for implementation guidance and code examples, andWallet Connect issuance for reference.

For wallet-bound flows, please see the pages onwallet-bound credential exchange flow for implementation guidance and code examples, andwallet-bound issuance mechanics andwallet-bound issuance service setup for reference.

Once you have clear your use-cases and your high-level architecture, you arrive at the build-or-buy decision. Verifying Verite credentials at scale and in production can be a massive undertaking if you're not familiar with OIDC token, authorization issues, applied cryptography, or other related problems in web engineering. That said, they are just signed JSON! If you're considering building a verifier for your environment, start with the tutorials and documents that guide you through the process in the"For Developers" section of this site .

If outsourcing this layer is more appealing, there are already a number of verifiers that offer the verification and validation/initial-processing of Verite credentials as a service, which can deliver definitive answers about which wallets/address to sort into which buckets or trust-levels over APIs, oracles, and/or on-chain registries. Some of these are even free to use in production, at scale! Updated5 months ago * Table of Contents * * Landscape * * Use-Cases * * Data Minimization * * Liability and Auditing Considerations * * Wallet Overview * * Architectural options