Overview

Suggest Edits

Landscape

In the world of cryptocurrency and web3, wallets are always walking a tightrope between adding new features and minimizing liabilities (security, maintenance, reputation). You want to incorporate features and interfaces that bring new value to your users (and more or more valuable users to your wallet); but you also need to avoid sticking your neck out and being the first (or only) wallet to introduce a new attack vector. You also have to balance engineering efforts and maintenance liabilities for any new feature-- it might start as a bespoke integration or experimental branch, but over time you are hoping the interfaces harmonize with others and standards emerge for shared security and reasonable debugging.

So where do verifiable credentials fit in? What is the reasonable engineering path for giving your users direct control over sensitive off-chain data and verifiable consent capabilities?

Some key questions to ask yourself:

1. What use-cases do you want to start with? Overview of use-cases
2. Do your users (or a reliably-gated subset of them) have ways to verifiably prove exclusive control? Identity Assurance
3. What is your users' and user-cases' tolerance for extending the wallet? Extensability and User Trust
4. Where does Verite sit in your more exotic or complex wallet configurations? Architectural options
5. Are you interested in supporting general-purpose Verifiable Credentials, and a Verite profile within that, or would you rather start small with ONLY Verite credentials? Supporting Verite versus Supporting All Verifiable Credentials
6. Interested in starting even smaller? Some Verite products facilitate Easy Integration Paths
7. , with full SDKs coming soon.

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 transactions involving 1 non-custodial wallet)
4. Currency controls/FX reporting and CBDC use-cases
5. Verifiable credit-assessment and forensic-sourcing

Identity Assurance

Different use-cases have different requirements and risk tolerances for "identity assurance," i.e., the degree of certainty or verifiability that the wallet is being controlled by the expected and authorized entity (usually a human). If control of a wallet can be transfered as easily as a private key in plaintext or a recovery phrase, your wallet's identity assurance is probably inadequate to most Verite use-cases without including some kind of third-party identity-assurance supplement (e.g. a "liveness API" from an IDV business that has previously enrolled the controller of the wallet). Verite works best with wallets that support on-device biometric unlock and/or hardware signing as an authentication factor to minimize the risks associated with stolen, borrowed, rented, or sold wallets.

Extensability and User Trust

Finding the simplest engineering path to meet the required non-technical requirements is often complicated by user experience, and specifically user tolerance for extending their software or altering their user experience. If your users are comfortable opting into an "experimental" branch of software, or using a "private build" of a provisioned/custom wallet, this is a lesser consideration; if they have to apply browser extensions or wallet plug-ins, this can drastically change the business calculus. For this reason, Verite is working on multiple form-factors and integration points in parallel, to give wallets multiple options.

Architectural Options

In today's crypto space, wallets are not homogeneous; the division of concerns between cryptography, [confidential per-user] storage, multi-device sync, and key custody can lead to complex arrangements of outsourcing and white-labeling. A

solution that only works for monadic wallets would not have legs in 2022, let alone 2025.

Verite's handling of verifiable credentials is best understood as a User-Experience capability, since wallet users have to understand the redirections and reliance between parties that leads them to "accept credentials" into their wallet and consent to their sharing with various front-ends (and, some day, directly with on-chain entities).

Storage of Verifiable Credentials

The Verite standards strive to remain maximally unopinionated and flexible about storage, as wallets today hardly have a standardized way of handling file storage in-wallet or in vaults. We have prototyped both a "split wallet" design, where verifiable credentials and user consent to shared are handled by an entirely separate piece of software (i.e., a device-specific browser-extension for conveying identity data) and a "minimum viable hybrid wallet", a cryptocurrency wallet which only handles Verite credentials in their simplest form in instance/device-specific local storage. Storage-supplemental dapps and PWAs are also under prototyping efforts in parallel, to allow an unmodified, traditional Wallet-Connect wallet to spin up a cross-device "vault" for verifiable credentials, with delegated signing rights over its contents for cleaner audit trails. We look forward to publishing the results of this research as replicable and even standardizable primitives soon.

Key Custody

Similarly, we have worked hard to keep direct access to keys optional to allow the kinds of custodial arrangements, including multi-party-computation and multi-signatory wallet architectures that are increasingly central to institutional handling of digital assets and complex custody/key-management flows. This means that wallets with supplemental hardware for signing or wallets utilizing multi-signature or multiparty-computation circuits are at no disadvantage. Over time, we believe the growing complexity abstracted by the concept of a wallet may well be the tailwind needed to standardize on wallet identity at a higher layer, e.g. with Decentralized Identifiers.

Verite-Support-versus-VC-Support

Verite could be the first you're hearing about Verifiable Credentials, or Verite could be one use-case in a broader Verifiable Credential strategy you've already been researching. Verite implementers to date have fallen at various points on the spectrum between these two poles, so we try to stay honest about the complexity of incrementalizing these efforts.

For implementers interested primarily or exclusively in Verite, we generally advise a "minimum viable complexity" approach: if you are targeting one or two use cases as an end unto itself, you can greatly simplify the implementation by thinking of Verifiable Credentials as a JWT you get from one domain and send to another with user consent. Much of the complexity of filtering and handling VCs differently by type can be hard-coded; much of the complexity of validating issuers can be outsourced to commercial or public verifiers, which allow this to parameterized and rely on registries of trusted issuers.

At the other end of the spectrum, it could be argued that implementing VCs today requires something like a "choice of stack". Verifiable Credential tooling is still in a somewhat early phase of its adoption; while there are ways to cross-verify and translate between different "flavors" of Verifiable Credentials, most open-source implementations and tools available today have one or two "native" verifiable credential encoding forms and secondary support for others. That said, Verite is built to what we believe to be the more promising and backwards-compatible stacks, all of which is open-source and most of which is governed and iterated through theDecentralized Identity Foundation . Wallets that handle JWT Verifiable Credentials with JSON Schema semantics get access to many emerging ecosystems and open platforms like Microsoft's Entra credentialing platform, the "Web5" efforts led by Block, employment credentials from Workday and Affinidi, etc. Open-source tooling from Spruce Systems, TBD, Transmute, and ConsenSys are good places to start if you are interested in general-purpose Verifiable Credential tooling that can be easily configured to handle Verite Credentials and turnkey revocation/status publication infrastructure.

Easy Integration Paths

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.

To make this more concrete, we have provided examples of simple and compound address-bound credentials. Click here for an example of wallet-bound credential, meant to be issued to and "presented" verifiably by a DID-capable wallet.

Liability and Auditing Considerations

Wallets do not hold outsized responsibility in Verite, thanks to the tamper-proofing inherent in Verifiable Credentials (integrity checks can be performed at issuance and again at presentation using standard JWT tooling, since Verite VCs are simply garden-variety JWT tokens).

Few Verifiable Credentials use-cases necessitate logging of VC events. Depending on the credential-type, though,

composite wallets (i.e., wallet integrating both crypto capabilities and identity/credential-management features) may want to log issuance events, particularly for credentials with a short shelf-life that need to be periodically reissued; more generally, they may want to filter verifiable credentials and handle them differently by type and/or by issuer.

Composite wallets, on the other hand, may well need more involved logging to vouchsafe their internal separation of concerns. For example, if a crypto wallet delegates identity functions or signing rights to a dapp, both dapp and wallet should log the delegation event in a replayable compact form (our lightweight implementation models this in the form of CACAO receipts). Similarly, if verifiable credentials are stored anywhere outside the controlling wallet proper, this delegation (and storage events and perhaps even retrieval events over which the wallet has signed) should be logged. Updated5 months ago * Table of Contents * * Landscape * * Use-Cases * * Identity Assurance * * Extensability and User Trust * * Architectural Options * * * Storage of Verifiable Credentials * * * Key Custody * * Verite-Support-versus-VC-Support * * Easy Integration Paths * * Liability and Auditing Considerations