REBOOTING THE WEB OF TRUST

DESIGNING THE FUTURE OF DECENTRALIZED SELF-SOVEREIGN IDENTITY

A WHITE PAPER FROM RWOT XI: THE HAGUE

On-chain Identity Proof Verification Design Patterns: An Evaluation of Different Strategies to Inject Digital Identity into Decentralized Applications

by Juan Caballero (https://Centre.io/), Martin Riedel (https://identity.com/), Egidio Casati (https://www.nymlab.it/), Robert Mao (https://ArcBlock.io/), Fabrice Rochette (https://2060.io/), Andrea Scorza (https://LTOnetwork.com/), and Raphael Roullet (https://violet.co/)

Abstract

In recent months, a problem space has been roughly delineated and widely discussed under the rubric of "on-chain web3 identity." Marketing imperatives and the lack of familiarity with compliance and liability issues particular to identity have largely muddied the waters in the solution space, however, resulting in a reductive three-way rivalry between "soul-bound tokens", verifiable credentials, and loosely-defined "zero knowledge" solutions. Working at a high level, we tried to identify a taxonomy that would be more useful for mapping this solution space according to high-level patterns and tradeoffs.

We defined the problem space at the highest level thusly: various architectures deploy on-chain artefacts to aid in the verification of claims about a wallet's controller. From there, we tried to bucket these into patterns before identifying strengths and weaknesses of each against a short, exemplary list of use-cases. The goal was not so much *evaluating* these exhaustively, as any evaluation should be more squarely grounded in more detailed use-cases and non- technical requirements. Instead, we strove to identify traits inherent to each high-level pattern that could lead to high-level fitness-for-purpose evaluations, i.e. the "strengths and weaknesses" of each.

Historical Context

On August 8th, 2022, the US Department of Treasury applied sanctions law as directly as it could against a decentralized application (dApp), by bringing the liability for interactions with OFAC-sanctioned entities to all Ethereum wallets that interacted with the Tornando Cash on-chain mixer. This precedent rightly inflamed concerns about the scope of national compliance, governmental overreach, and protocol neutrality.

This precedent presents a once-in-a-lifetime opportunity to leverage on-chain digital identity as a way to anticipate the regulation trajectory and unlock adoption of decentralized applications (including decentralized finance) with sophisticated compliance and reporting/auditing strategies. The impetus for this paper was largely to analyze different ways of verifying and consuming identity credentials on-chain as part of a process for validating, gating, or accepting transactions involving digital assets (i.e. "DeFi transactions").

To make an architectural reference comparison that would be more useful to a broader audience, however, we abstracted this focus a bit. This enabled us to re-scope the subject of inquiry to a comparison of architectures where a fundamentally pseudonymous, decentralized PKI system (i.e., a "blockchain wallet"-driven architecture) could be used to prove and verify real-world claims about the end-users controlling agents (i.e., wallets) in that system without necessarily deanonymizing those end-users in the immutable records left by a transaction.

Methodology

Our analysis considered diverse solutions with different privacy strategies, ranging from on-chain tokenized identities (i.e. "Soulbound Tokens") to on-chain validation of proofs derived from correlation-resistant verifiable credentials (based on the `anonCreds` system of zero-knowledge presentation).

This paper strives to be informative as high-level overview and introduction for the following audiences: - Originators of on-chain assets, i.e. minters and issuers - Resellers of on-chain assets or operators of marketplaces and exchanges - DAOs and other on-chain communities - Policy-makers and regulators surveying technical possibilities for compliance frameworks for on-chain assets - Developers of "decentralized apps" (dApps) and other on-chain applications

This paper isn't intended for asset holders or end-users in general, as we expect they should benefit from sound regulation unlocking mass adoption via usability, while preserving on-chain privacy and good-faith implementation of such compliance frameworks.

Terminology

Because the solution space for this problem is fresh and terminology is still emerging, it was important for our group to align on a common nomenclature within the scope of the analysis. This summarizes our aligned usage:

Asset Smart Contract

- AKA: Token, Fungible Token, Non-Fungible Token, Composite Token
- Def: On-chain building block that governs ownership and transfers of an on-chain asset such as a stablecoin, an NFT, etc. These often predate dApp smart contracts and are controlled separately.

Credential Issuer

- AKA: Issuer (context sensitive)
- Def: Party that issues real-world information about a holder in a verifiable form, whether that form be on- or off-chain (i.e. badges or verifiable credentials, respectively); it issues these directly into the control of the holder (in these use-cases, the holder can be assumed to be the data subject; badges not burnable by the holder are deprecated and out-of-scope)
- Note: In DeFi and banking parlance, an "Issuer" usually refers to the issuer of a security or asset, often referred to as "tokenized" if represented by (or coequal to) an on-chain asset like an ERC-20 token. Our survey did not include "gated" or "compliant" on-chain assets, so all references to "issuance" here refer to identity tokens, whether on-chain or off-chain (i.e. verifiable credentials). See Asset Smart Contract.

dApp, i.e., Decentralized Application

- AKA: DeFi app, "protocol" (as in, "lending protocol" or "DEX protocol")
- Def: 1 or more front-end and 1 or more decentralized protocols such as on-chain smart contracts, IPFS, or P2P protocols, interact with deployed contracts and/or chain state

dApp front-end

- AKA: Decentralized front-end, DeFi front-end
- Def: A user interface that communicates with 1 or more decentralized protocols such as on-chain smart contracts, IPFS, or other P2P protocols

dApp Smart Contract

- AKA: On-chain program (Solana), Chain code (Hyperledger), or similar concepts etc.

- Def: A deployed piece of persistent code executing on a blockchain virtual machine environment, regardless of EVM-compatibility, L1/L2 structure, etc. This is sometimes modeled as the "on-chain component" of a dApp, which executes business logic and results in on-chain state changes (e.g. a DEX contract, a lending protocol, etc.)

Data Custodian

- AKA: Data Controller (GDPR), Identity Provider (OIDC), KYC Custodian
- Def: Storer and controller of data needed to verify identity of an actor (can be internal or outsourced). Some wallets interact directly with a semi- or self-custodial per-user data store (encrypted data vault, decentralized web node, personal data store, locker, etc) which can also be considered custodial in a technical sense, although more mapping to a more regulated custodial actor via complex authorization system(s) may be needed for some regulated use cases that specify legally the definition and obligations of a custodial role for data per se.

Gated Access

- AKA: Token-gating
- Def: Restricting access and/or providing exclusive content, right or membership to some kind of service. In the dApp/DeFi context, this can refer to "front-end gating" (access control in the browser to a web resource regardless of origin) or "back-end gating" (wallet checks done by smart-contracts on originating/signing accounts)

Gatekeeper

- AKA: Gatekeeper (Identity.com) Verifier (Verite), White-lister (Aave Arc), Identity Oracle, Validator Node (ArcBlock), Proofi (LTONetwork)
- Def: The enforcer of a "checkpoint" for actors in a system, in this case accounts in a pseudonymous asset system like a blockchain (whether UTXO-based, account-based, or some other). This entity re-issues or bridges (possibly trustlessly) verifiable information into a closed trusted ecosystem like a blockchain.

On-chain Badge

- AKA: Gateway Pass or Gateway Token (Identity.com), Human-Bound Token (violet.co), Opaque Identity Token (KYCDAO), Passport NFT(ArcBlock), or Badge Token (Swirlds Labs)
- Def: A verifiable on-chain badge includes one or more claims made by an authoritative/known issuer minting/publishing those badges. Note that this has no relation to OpenBadges qua specified technical system. Badges can be minted by Credential Issuers, or by Gatekeepers that are also Credential Issuers, or by Gatekeepers "bridging" claims issued off-chain by Credential Issuers.

On-chain Relationship Token

- AKA: Sismo token, ENS NFT, etc.
- Def: An opaque on-chain artefact identifying an account as having a relationship with an authoritative issuer, data custodian, etc. Unlike a badge token, these do not convey any claim about the bearer other than a relationship relative to the token's issuer, which has to be queried separately (via API, on-chain, or otherwise) for claims about the subject.

Architecture A: Public Badge Token directly represent real-world claims (badge-only)

Contracts can directly interpret on-chain badges to make decisions (Web 2.0 authentication or Web3 transaction authorization). This specifically can include information that is found within the Token itself. For example, all NFTs on a given blockchain containing a specific trait are allowed to execute an instruction on a specific smart contract. No additional calls to other data sources (elsewhere on-chain or off-chain) are required to accept or reject a transaction or make authentication decisions.

While multiple ERC draft standards are being used, these are generally referred to in the market and in blockchain discussions as "access/gating based on Soul-bound Tokens (SBTs) or Account-bound Tokens, i.e. an ERC721 without a transfer function interpreted as a badge representing a claim.

Replayability

Since all data is on-chain and fully public, all logic can be replayed or audited on-chain.

Diagram

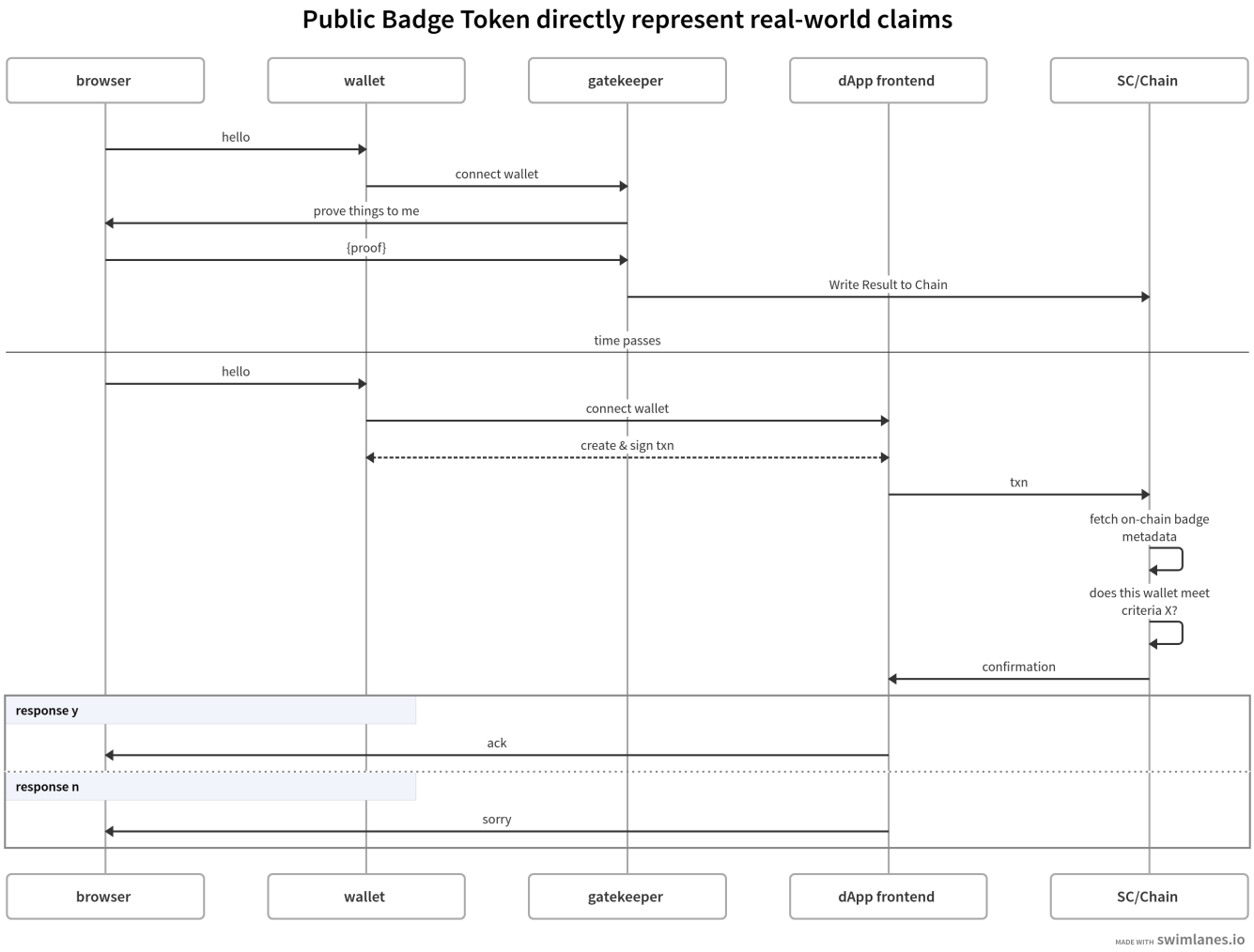

diagram of architecture A from https://swimlanes.io/d/EVjTr690X?e

Architecture B: Opaque Relationship Token triggers query to intermediary (trusted intermediary)

In order to minimize identifying information in relation to the public approach, one solution is to minimize the exposed information to a well-defined minimum. In this sense the access decisions are still made by a smart contract combining badge metadata with additional data fetched just before and submitted at time of decision, whether that additional data comes from:

1. an API call from a dApp front-end to issuer or other intermediary,
2. an oracle call,
3. another smart contract, or
4. a valid transaction signed by the issuer or other trusted intermediary and delivered via the wallet.

In this approach, the later/just-in-time call (which might be “expensive” computationally or in gas/transaction fee terms) is validated along with the first; any claims implied by the relationship token (status at time of issuance) are not considered valid until confirmed or updated (status at time of last-minute query).

Replayability

Auditing and replaying transactions that involve a second data source pose real issues for these architectures: at a later time (long after transaction), the second-data-source call needs to be verifiable and/or replayable, at least in combination with on-chain records. In some sense, the trust model and security model is the "worst of two worlds" of the two data sources. For example,

1. An API must allow "historical queries," i.e. the API call needs to be replayable and verifiable after the underlying state changes (which changes the trust model for some use-cases). One way of strengthening this replay on a technical level is to include current `block_height` or other ledger-time information in query and response. Historical query could also be protected or authorized per-wallet by an off-chain signature from the wallet, i.e. a CACAO session receipt.
2. Oracles must similarly maintain and verify historical state in the case of future replay. (Most oracles log events to on-chain records by default)
3. Dependencies on additional smart contracts increase code-auditing surface and trust threshold.
4. If the second data source passes information to the smart contract validating or generating the gated transaction in the form of an off-chain transaction (a message that could be written to the chain by the first smart contract but is not intended to be executed), at least one of the two actors (data source and/or smart contract) needs to store and access this non-executed transaction for the entire flow to be replayable, and both may need to store some kind of record identifier to insure end-to-end completeness of the records.

Diagram

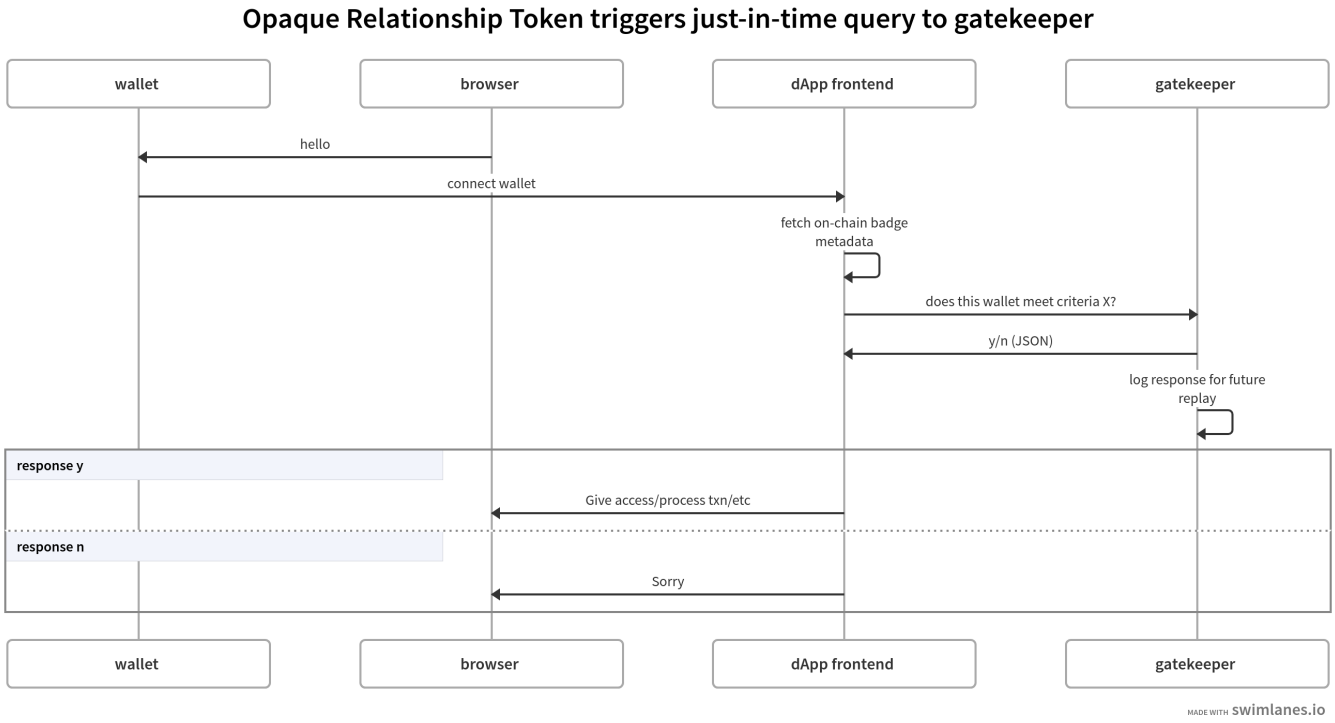

diagram of architecture B from https://swimlanes.io/d/3ZXqzGe0h?e

Architecture C: Blinded Off-Chain Proof Verified On-Chain

Smart Contracts receive a ZK proof generated elsewhere (i.e. off-chain, in- wallet, in-dApp, etc), which is verifiably derived from a previously issued credential. The proof is generated by the credential holder within the bounds of the ZK framework, proofing one or more predicates and/or attributes about the credential against a static credential schema. (Note: not all verifiable credentials are capable of and useful to these kinds of proofs; the VCWG has yet to specified standardized data

models for these credential schemata and requisite credential proofing mechanisms such as per-property signatures, etc.; for ongoing incubation work on protocols, see the Applied Cryptography working group at DIF, Blockchain Commons, and other community groups).

A dApp-specific smart contract receives the given proof and verifies its correctness (against a known and/or static credential schema) and determines whether the transaction can be effectively processed or not. This is often referred to as an "Anonymous Credential" (because it relies on a correlation- resistant credential rather than one authenticated by direct reference to a static public identifier). The authors note that it would be more precise in this use-case to refer to it as a "Fully pseudonymous proof" because it links an anonymous credential to a pseudonymous identifier (i.e. a blockchain account). Note the division of labor between application-specific verification (against application-specific compliance requirements) in a distinct smart contract from the underlying asset contract; this is typical of many regulated assets.

Replayability

The full pseudonymity of the proof presented to a verifier requires the verifier to trust the ZK proof generation and the [potentially unknown] issuer of the underlying credentials (depending on schema and proofing mechanism used). Since minimizing logic on-chain is often a constraint or design goal of these systems, "trust establishment" (deciding which issuers are adequate at time of proofing and/or at time of verification) may introduce an additional intermediary and/or record-keeping obligation. Replaying the entire flow requires the proofing mechanism and/or schema to include record locators (or at least routing information on how to get the relevant records from the parties involved).

Diagram

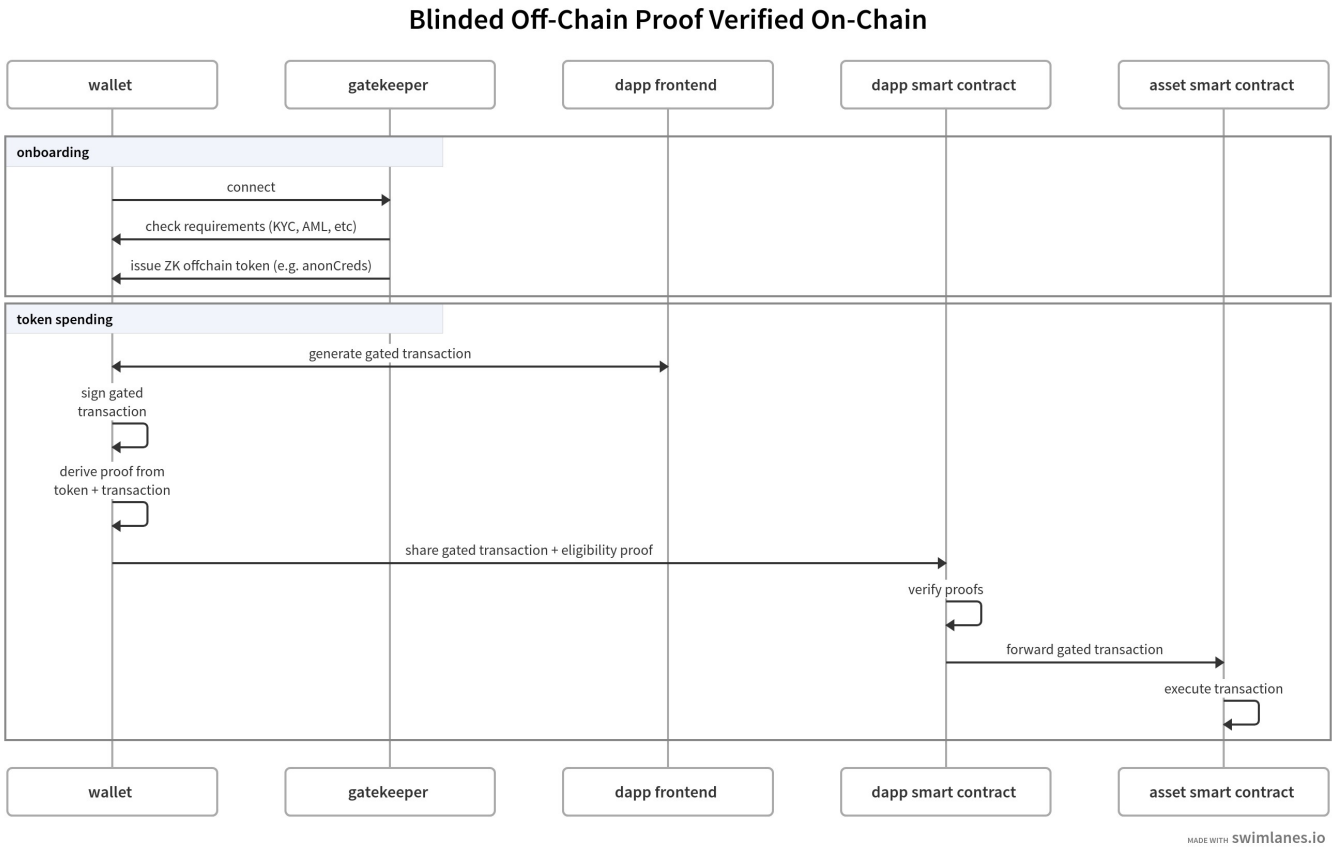

diagram of architecture C from https://swimlanes.io/d/8p8Ua7zK9?e

Architecture D: P2P Off-Chain Verification

All of the architectures above include an implicit client/server hierarchy: end- users relate via agents to one or more on-chain protocols via one or more web front-ends, i.e.:

```
end-user > wallet/agent > web front-end(s) > smart contract backend(s)
```

There are, however, many peer-to-peer use-cases that we can imagine enabling with wallet/agents and/or websites mediating and assisting. Indeed, if the concept of a Data Custodian ends up established in the market and in policy, we can assume custodians will want to branch out into the peer-to-peer space, supporting trust establishment and "gating" for peer-to-peer payments in both the self-custodial and custodial space, the latter currently dominated by KYC-obligated centralized exchanges.

For example, imagine a classic P2P user story: - Alice wants to send funds to Bob. - Alice sends Bob a signed transaction authorizing a transfer to Bob. - **Before deciding whether or not to forward this valid transaction** to the blockchain, thus executing a balance update to the relevant Asset Smart Contract for all the world to see, Bob would like to validate Alice according to some criteria. - Bob requests a proof against that criteria (or Bob already received it with the authorization). - Bob verifies proof and determines whether to accept the transaction and forward it on to the chain.

The authors compared use-cases that they see in today's web3 economy, from verifying funds paid by a decentralized service, to DAO funding (proving legitimate representation of a DAO by a given actor or address), to DAO or bounty disbursements, to interactions with unknown legal organizations in unknown jurisdictions, where *negative* proof of a jurisdiction may be adequate to proceed. Comparing these real-world demands, we proposed a hypothetical high- level architecture that abstracts out the specifics of Alice and Bob's software agents and the specifics of their data custodians and proofing mechanisms. Inheriting the proofing assumptions and agent architectures of the other three architectures analyzed would yield three different lower-level versions of this architecture, but we opted to keep the simplicity of the higher-level abstraction for the sake of our comparison given that unlike the other three, there are no products fitting this pattern in production today.

Replayability

As in FATF use-cases, fully P2P use-cases can be the hardest from which to produce and guarantee adequate record-keeping. The authors remark that perhaps per-user data stores or custodial services could conceivably emerge in the coming years that process, store, and make queryable records of such off-chain interactions and proofs.

Diagram

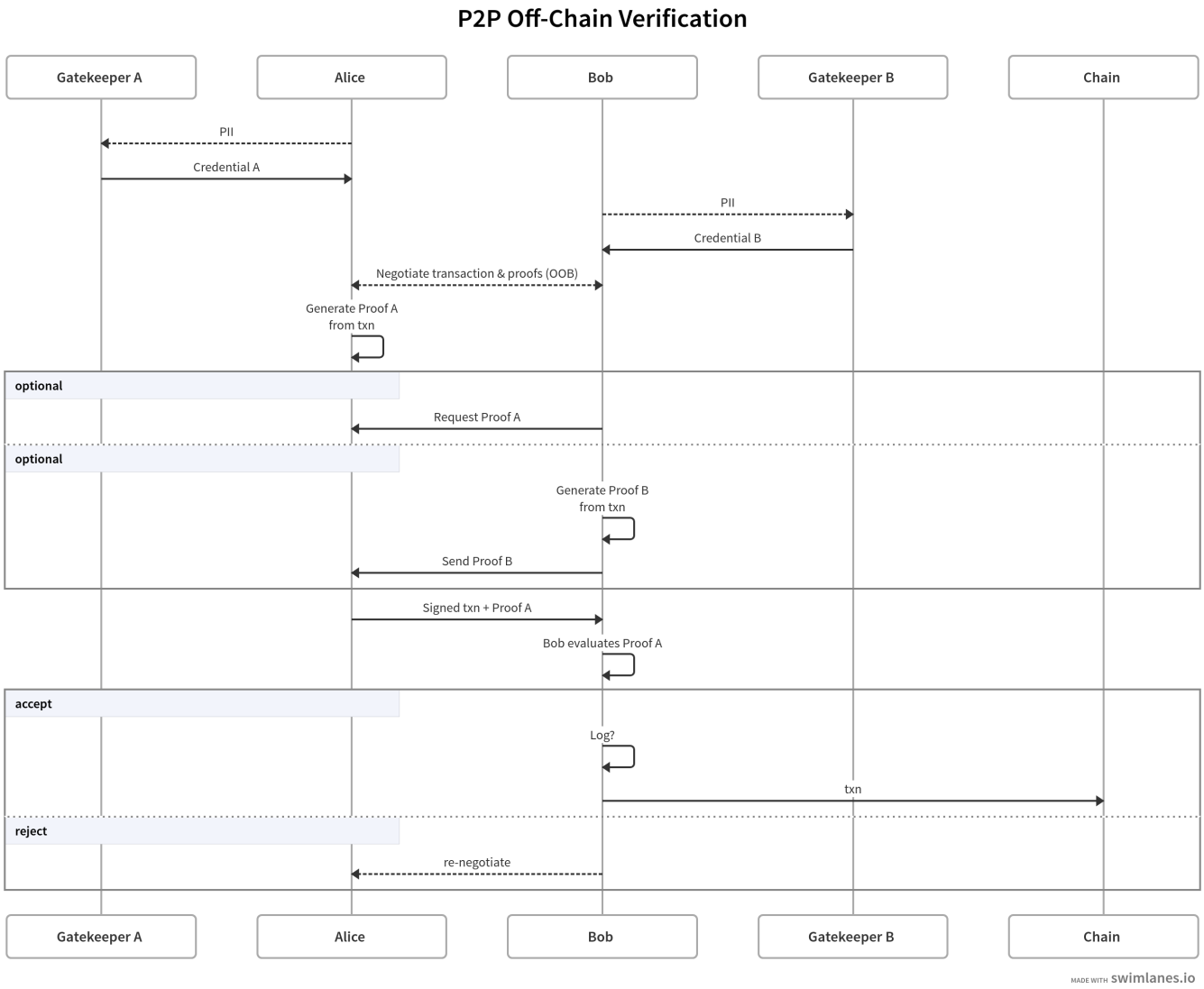

diagram of architecture D from https://swimlanes.io/d/84YZKRttH?e

Use cases

1. Regulated protocol would like to restrict or gate access to a smart contract (or even just to a dApp frontend) to wallets that have been cleared against a predetermined wallet-controller criterion relevant to its regulators
    - Example criteria: Only wallets whose controllers have been KYC-onboarded by a trusted KYC custodian
    - Real-world examples: Verite.id ecosystem issuers like Circle and Violet; KYCDAO; aka “KYC-gated DeFi”; Identity.com’s Gateway Protocol
    - Auditing requirement: some smart-contract interactions require all parties trusting/relying on a custodian to be able to query custodian for underlying PII in case of subpoena, etc.
2. Allowlist/denylist for NFT mint events - Regulated criteria encoded as offline VC, checked before by front-end at time of mint transaction
    - Example criteria: “account has never transacted with Tornado Cash” or “account not flagged as suspicious by X, Y or Z machine-learning algorithm trusted by regulators”
    - Real-world examples: see guest Blog Post by Justin Hunter (Pinata)
3. Tokenized security would like contract-level enforcement of wallet-holder requirements to guarantee ledger-wide compliance
    - Example requirements: wallet controller must be KYC’d or KYB’d with known jurisdictional competent authority and, if required, have verifiable affidavit of Investor Accreditation
    - Variant: only transact SALES (or only transact TRANSFER methods) to wallets that can deliver verifiable proof to the smart contract at time of any sale or transfer; enforced by EIP712 or other metatransaction/multisig mechanism requiring proof to be checked before transaction is accepted
    - Less popular variant: “interventionist” smart contract which can freeze, revoke, or seize asset held by non-conformant (or sanctioned) wallet aka “SEC accredited investor” (for fund raising)

High-Level Analysis & Conclusions

We grouped architectures into four high-level categories and analyzed them. Here is a high-level summary of relative use-case fits and challenges.

Properties Chart

Architecture	How Reconstruct Audit Trail	Trusted Intermediaries	Simplicity	Implementation Complexity	Flexibility
A	On-chain receipts + Issuer records	None	High	Low	Low
B	All parties (Issuer & Verifier) must be subpoenaed/ cooperate	Verifier must be trusted by relying party and wallet controller	Low	High	High
C	Issuer & Wallet Controller need to cooperate to reconstruct (poss. also circuit-building middleware)	Whoever combines ZK inputs into a “circuit” needs to be trust by all	Medium	Medium (depends on SDKs & middleware used)	Medium
D	Issuer & 2 Wallet Controllers need to cooperate to reconstruct	Routing/Messaging infra, possibly also proofing/dereferencing support for complex VCs	High	High (until DIDComm is more mature?)	High

Same Info, Axis-Flipped

Properties	Arch A	Arch B	Arch C	Arch D
Audit Trail	On-chain receipts + Issuer records	All parties (Issuer & Verifier) must be subpoenaed/cooperate	Issuer & Wallet Controller need to cooperate to reconstruct (poss. also circuit-building middleware)	Issuer & 2 Wallet Controllers need to cooperate to reconstruct
Trusted Intermediaries	Whoever combines ZK inputs into a “circuit” needs to be trust by all	Routing/Messaging infra, possibly also proofing/dereferencing support for complex VCs		
Simplicity	High	Low	Medium	High
Implementation Complexity	Low	High	Medium (depends on SDKs & middleware used)	High (until DIDComm is more mature?)
Flexibility	Low	High	Medium	High

Architecture/Use Cases Applicability

Use Case	Arch A: Badge Token	Arch B: Identity Token + Intermediary	Arch C: On-Chain Verification	Arch D: P2P Verification
1	Fastest to implement	Most reasonable for compliance in today's EVM	Verification-capable VMs and business models emerging	Best for single-jurisdiction/single-criterion minting contexts and E2E single-vendor use-cases
2	1:N relationship of badge token to verifiers difficult unless verifiers harmonize requirements/logic, which is currently very unrealistic	May be too complex for today's use-cases, e.g. NFT market	May be too technically demanding for today's NFT industry and use-cases	Depends on business & regulatory model of NFT mint context
3	Moving beyond API-based architecture may require as-yet theoretical tooling (i.e. Oracle connection to verifier/intermediary)	Good building block of a compliance-built-in stack/VM/etc	Possibly a good fit for FATF-conformant reporting, esp. for self-custody wallets, if FATF protocols enable P2P/self-custody solutions	

Same Info, Axis-Flipped

Architecture	Use case 1	Use Case 2	Use Case 3
A: Badge Token	Fastest to implement	Best for single-jurisdiction/single-criterion minting contexts and E2E single-vendor use-cases	1:N relationship of badge token to verifiers difficult unless verifiers harmonize requirements/logic, which is currently very unrealistic
B: Identity Token + Intermediary	Most reasonable for compliance in today's EVM	May be too complex for today's use-cases, e.g. NFT market	Moving beyond API-based architecture may require as-yet theoretical tooling (i.e. Oracle connection to verifier/intermediary)
C: On-Chain Verification	Verification-capable VMs and business models emerging	May be too technically demanding for today's NFT industry and use-cases	Good building block of a compliance-built-in stack/VM/etc
D: P2P Verification	Somewhat unrealistic short-term	Depends on business & regulatory model of NFT mint context	Possibly a good fit for FATF-conformant reporting, esp. for self-custody wallets, if FATF protocols enable P2P/self-custody solutions

Further Research Directions

All of the above assumes verifiable credentials to be inherently off-chain artefacts, although there is plenty of research on smart-contract issuance and verification of verifiable contracts. In particular, NymLab has been prototyping (and even demoed at the Rebooting Web of Trust 11 where we held our face-to-face discussion for this paper) an implementation of AnonCreds verification on a custom Cosmos runtime, which can be passed configuration variables and a credential to trigger other smart-contract functions. As this space evolves and matures, it will surely add more architectures to this comparison. See also Violet's prototype for on-chain verification on today's Ethereum Virtual Machine with hard-coded configuration, constrained to VC-JWT's that identify issuers and subjects by Ethereum addresses.

Another extension to the model of this analysis comes from the distributed cross-chain/multi-chain authentication represented by the evolving CACAO authorization-receipt format being worked on in the CASA Community. This model takes wallet signing methods used today for off-chain purposes and profiles them to create a "Sign In with X" flow, where X is a wallet signature standard specific to a given blockchain ecosystem. While this might appear to be a copycat to Web2's "Single Sign-On" pattern, this *also* a building block of authorization as well, since it allows a payment private key to delegate identity functions to an ephemeral or custodial key which could, in turn, be used to create proofs, present verifiable credentials, or otherwise engage in protocols useful to the identity purposes of on-chain actors. With time, the authors expect significant new architectural options to arise from this kind of delegation and protocol bridging.

Another simplification above is that addresses were considered as monadic and isolated, with a 1:1 relationship between wallets and addresses. In practice, however, blockchains vary greatly in their support for multi-address wallets and accounts, with varying degrees of "Account Abstraction" implemented at the protocol layer to allow end-users to selectively disclose co-control of multiple addresses. These capabilities are central to Identity.com's design for did:sol (the Solana on-chain DID method), which enables Solana wallets to use a secret, unfunded address as a credentialSubject for VCs without linking them to other addresses the same user publicly uses on-chain.

As Account Abstraction patterns become more dominant, particularly as it gets enabled on the protocol level by Ethereum and EVM-compatible chains in the coming months, we can expect this complication to cross-cut the above patterns and lead to a more complex architectural landscape. For example, cross-chain capabilities enabled across multiple account-abstraction-enabled chains will complicate and advance cross-chain engineering substantially. Today, Identity.com provides an on-chain permissioned Token implementation based on their Gateway Protocol design with implementations on both Solana and EVM. In this way, did:sol allows its holders to link keys from both environments (`ed25519` and `secp256k1`); as such patterns become more common, the authors expect the privacy landscape could change as much as the economic landscape will.

One factor elided from our analysis above was consent– crucial to the product design of Violet.co's Humanbound token is the consent of the holder to minting, which is revocable by both issuer-initiated OR holder- initiated burning. When evaluating specific products rather than high-level architectures, architects and policy researchers alike are strongly encouraged to consider meaningful consent and end-user terms of service in any taxonomy of data custodian relationships and flows.

References:

- Soulbound Whitepaper
- DeSoc Whitepaper
- How To Use Verifiable Credentials And Verite To Build An Off-Chain NFT Allowlist (Verite docs)
- Pre-print draft of Chainlink Labs Research on ZK Circuit-based Verification (iacr / Chainlink)
- NFT metadata fragility (the verge)
- MetaMask snap for VC/VP handling in crypto wallets (medium / BlockchainLAB UM)
- Proof-of-concept for verifying VCs on-chain on EVM chains (Violet.co research)

Additional Credits

Lead Author: Juan Caballero

Authors: Juan Caballero (Centre.io), Martin Riedel (identity.com), Egidio Casati (NymLab.it), Robert Mao (ArcBlock.io), Fabrice Rochette (2060.io), Andrea Scorza (LTOnetwork.com), Raphael Roullet (Violet.co)

Sample APA Citation

Caballero, J., Riedel, M., Casati, E., Mao, R., Rochette, F., Scorza, A., and Scorza, R. (2023). On-chain Identity Proof Verification Design Patterns. Rebooting the Web of Trust XI. Retrieved from https://github.com/WebOfTrustInfo/rwot11-the-hague/blob/master/final-documents/onchain_identity_verification_flows.pdf

About Rebooting the Web of Trust

This paper was produced as part of the Rebooting the Web of Trust XI design workshop. On September 26th to 30th, 2022, over 60 tech visionaries came together in The Hague, The Netherlands to talk about the future of decentralized trust on the internet with the goal of writing at least 5 white papers and specs. This is one of them.

• **RWOT Board of Directors:** Christopher Allen, Joe Andrieu, Erica Connell.

• **RWOT11 Coordination Team:** Will Abramson, Christopher Allen, Joe Andrieu, Shannon Appelcline, Erica Connell, Eric Schuh, Carsten Stöcker.

• **Workshop Credits:** Will Abramson (Producer), Christopher Allen (Founder), Shannon Appelcline (Editor-in-Chief), Erica Connell (Host), Amy Guy (Ombudsperson), Willemijn Lambert (Graphic Recorder), Eric Schuh (Ombudsperson), Carsten Stöcker (Co-Producer, Demo Organizer), Dorothy Zablah (Facilitator).

• **Gold Sponsors:** The City of the Hague, Digital Contract Design, Dutch Blockchain Coalition, The Hague University of Applied Sciences, eSSIF-Lab.

• **Contributing Sponsors:** Blockchain Commons, Legendary Requirements, Spherity.

Thanks to all our attendees and other contributors!

What's Next?

The design workshop and this paper are just starting points for Rebooting the Web of Trust. If you have any comments, thoughts, or expansions on this paper, please post them to our GitHub issues page:

https://github.com/WebOfTrustInfo/rwot11/issues

The twelfth Rebooting the Web of Trust design workshop is scheduled for September 18-22, 2023, in Champagne, Germany. Sign up for announcements at https://weboftrust.info/subscribe/. If you'd like to be involved or would like to help sponsor the event, email: Leadership@WebOfTrust.info