The Global Access Protocol

(A draft white paper)

The Digital Securities Initiative

The Global Access Protocol (GAP) facilitates the deployment of "Regulated Zones" that set and enforce rules that enable participants to be compliant. The base protocol is agnostic to any particular jurisdiction, but is designed to address a plethora of regulatory objectives such as money laundering, financial crime, and market integrity. Embedding financial regulations into the smart contract layer, GAP can be deployed on both public and private permissionless chains, enabling compliance while preserving the benefits of Decentral Finance (DeFi). It prioritizes open standards, composability, competitive service provider markets, and privacy, preventing value extraction or regulatory capture by inefficient intermediaries and ensures that tokenized assets are implemented in a way that DeFi ecosystems can integrate, rather than shaped to benefit any one company, platform, or blockchain.

1. Problem Statement

The emergence of permissionless smart contract blockchains and decentralized finance (DeFi) challenges many of the foundational assumptions of traditional financial regulation. These decentralized systems are governed by code and cryptographic consensus rather than by intermediaries or institutional discretion. Anyone can deploy applications, issue transactions, or operate infrastructure without requesting approval or providing identification.

In 2025, we are witnessing the merging of DeFi and traditional finance (TradFi). We see this merging through the proliferation of TradFi instruments which hold crypto assets (i.e. Bitcoin Exchange Traded Products), and conversely through the representation of regulated securities and other assets onchain, commonly referred to as the tokenization of real world assets (RWAs). In this document we consider the latter effort.

Regulatory barriers have been seen by many industry participants as a central impediment to the tokenization of RWAs[1][2]. In DeFi, conventional enforcement methods such as identity-based licensing, manual oversight, and centralized control do not function as they do in traditional financial systems due to the lack of centralized intermediaries. Moreover, lacking the typical intermediaries, the exact legal obligations of DeFi projects and users under the Bank Secrecy Act (BSA), Securities and Exchange acts, and other statutes are unclear, leaving the nascent DeFi space vulnerable to a capricious regulation.

However, these intermediaries require complex technical coordination and sophisticated regulatory frameworks, entrenching them and enabling them to extract high fees from markets. So by removing these intermediaries, DeFi streamlines trades and settlement, enabling instantaneous settlement. Furthermore, DeFi creates competitive service provider markets, where anyone can host infrastructure or provide liquidity, ensuring low transaction fees in trades and low borrowing costs. Most importantly, DeFi lowers the barrier of entry to investors, allowing a far more people to share in the wealth generated by capital markets. Thus, if these regulatory issues were resolved

[1]https://www.oecd.org/content/dam/oecd/en/publications/reports/2021/03/regulatory-approaches-to-the-tokenisation-of-assets_da7ae482/aea35466-en.pdf
[2]https://natlawreview.com/article/tokenization-real-world-assets-opportunities-challenges-and-path-ahead?utm_source=chatgpt.com

correctly, a new Global DeFi Market system would arise that would seamlessly bridge "normal" DeFi markets and global TradFi markets. The benefits of tokenization would usher in a golden age of frictionless finance where liquidity and capital would be accessible to everyone.

Taking cues from various regulators, some brave projects have begun to plot courses through these uncharted waters in an attempt to build the new global DeFi market structure. While these efforts are valiant, no project has offered a comprehensive vision for regulation that preserves the interoperable essence of DeFi. As a result, instead of a golden age of finance, we risk creating a fragmented patchwork of different regulatory solutions, or a single centralized solution swallowing DeFi, negating its core value proposition.

The Global Access Protocol (GAP) is a platform for defining and enforcing standards. This tool is designed to build configurable compliance frameworks on permissionless chains through which RWAs can be compliantly tokenized. The protocol itself is agnostic to any specific regulations. Instead, it provides a platform where rules are either encoded into smart contracts, or, if a machine is unable to reason about a rule, it is enforced by service providers via common ontological standards. More specifically, GAP provides a suite of contracts that allows administrators to deploy and administer a *Regulated Zone* in which participants can issue assets and use DeFi protocols. The administrator of each Regulated Zone defines rules via terms of service agreements which they can enforce via smart contracts or by ejecting participants operating under their purview.

Specifically, GAP is designed to address five specific problems.

1. **Whitelist Fragmentation** – *The proliferation of isolated token whitelists that do not interoperate.* Relying on each token or platform to maintain its own whitelist of eligible addresses leads to inefficiency and fractured liquidity, requiring users to go through KYC repeatedly and DeFi devs to continually seek approval on token issuer's whitelists and write bespoke integration code.

   GAP's Solution: With "One Stop" onboarding, users can access large segments of the market with a single onboarding experience. Specifically, administrators grant permissions to intermediaries who issue Verifiable Credentials granting access to Regulated Zones according to a specified standard. Thus each Regulatory Zone is an open, federated market. Furthermore, Intermediaries can operate in many different Regulated Zones, ensuring that users have credentials for a variety of assets and platforms, avoiding this fragmentation issue.

2. **Deglobalization Letdown** – *Fragmentation of markets along political and regulatory jurisdictions, breaking interoperability and undermining capital formation.* If each regulatory jurisdiction, either that of a nation state or of a specific government agency, forces assets to comply with bespoke regimes, capital will become siloed into networks, breaking the great promise of frictionless finance.

GAP's Solution: Regulatory zones live side by side on the same ledger, allowing interoperability. Specifically, since contracts and users can operate in multiple Regulatory Zones at once, assets and contracts can be composed via the overlapping zones. This approach emphasizes cross-jurisdiction interoperability, so compliance checks from multiple regulators (both domestic and international) can coexist. This prevents the deglobalization of markets and maximizes liquidity depth.

3. **Surveillance Nightmare** – *An Orwellian outcome where every transaction and identity is surveilled.* Without the proper protections, blockchains have the potential to become a mass surveillance tool. Aside from the obvious challenge to individual freedom, this state of affairs prevents investors from protecting their trading strategies and knowledge about their positions.

   GAP's Solution: Although GAP is not itself a privacy preserving technology, it can be composed with existing privacy technology to allow intransaction data and personal identities are only visible to accountable, regulated parties on a "need to know" basis. Moreover, sensitive data needs layers of privacy and to be segmented into discrete groups to protect against leaks and hacks.

4. **Doxing Slow Burn** – *Gradual erosion of pseudonymity over time through data leakage.* Even if personally identifiable information (PII) starts out hidden, poorly designed systems leak clues which can link wallet addresses to real identities. Over time, as each transaction essentially 'doxes' a user to their counterparty, a large percentage of wallet addresses de-anonymized, eliminating the semblance of privacy available onchain today.

   GAP's Solution: GAP itself does not directly link any two transactions with public information, and thus enables a user to obfuscate their transaction history from the general public. This is achieved via a system of pseudonyms that allow a user to compartmentalize their identity and reveal it on a need to know basis.. Even long-term participation in the regulated zone does not inevitably reveal a user's identity to the world.

5. **Regulatory Moats** - *When a small group of service providers use regulatory requirements to entrench themselves.* Regulations often manufacture demand for certain mandated services. If the barrier to entry for such a service market is too high, then competition is impossible and the system becomes dominated by an oligopoly, severely affecting the quality of service provided.

   GAP's Solution: All service provider markets are designed to be competitive. In particular, anyone can deploy a Regulated Zone ensuring a competitive and contestable market. Moreover, GAP can be deployed on permissionless infrastructure and preserves the traditional benefits of DeFI.

2. Summary of GAP

In the Global Access Protocol, a *Regulated Zone* is a collection of contracts that have implemented access controls according to a common standard under the auspices of a particular administrator. These access controls are implemented into a smart contract via a *Transaction Filter*, which checks whether an address has the permissions to be an argument of particular functions. A Transaction Filter does not just filter users, but it also controls how its contract can be composed with other contracts.

The Transaction Filter makes these determinations via common standards, specifically *Attribute Regimes* which encodes personal information about users; *Risk Flags* which indicate certain user transaction patterns; and *Classification Regimes* which classify smart contracts based on their functionality and legal obligations. The determinations made by Transaction Filters using these standards are deterministic and automated. However, human (or AI) assessments are required to apply these standards to actual users and contracts. These assessments are done by service providers operating on top of GAP, namely:

- *Identity Keepers* collect user personal information during onboarding and issue *Verifiable Credentials* with the appropriate Attribute Regime. These credentials grant access to Regulated Zones.
- *Transaction Monitors* monitor the blockchain and record all calls to a particular contract. Based on this data they use an *Onchain Risk Database* to issue Risk Flags when undesirable behavior is detected. These flags can revoke access to a Regulated Zone.
- *Contract Certifiers* inspect contracts and issue Verifiable Credentials according to a Classification Regime. With these credentials, a contract can be composed with other contracts within a Regulated Zone.

The ability for any Regulated Zone to achieve its regulatory objectives is limited by the efficacy of the Service Provider's methods. Thus these service providers will be supervised by two types of *Trust Anchors* that administer the regulatory zone. These Trust Anchors will set rules and standards with each of the above service providers by, for example, Terms of Service agreements. Trust Anchors can enforce these rules by ejecting offenders from their Regulatory Zone. As such, the Trust Anchors themselves are part of the standards utilized by the system: any Verifiable Credential indicates which Trust Anchor it was issued under, allowing participants to judge their veracity.

Composability requires counterparties to share a common standard for communicating. In the same way the ERC-20 and ERC-721 standards unlocked Ethereum's "money-lego" dynamic by

defining uniform interfaces for tokens and NFTs[3][4][5], an ontology governed by these Trust Anchors standardizes classification processes and risk so that credentials are portable across venues. This standardization creates a federated whitelist and avoids Whitelist Fragmentation, where each protocol maintains bespoke allowlists that fragment liquidity and reintroduce chokepoints. Traditional finance solved a similar problem with the FIX and ISO 20022 standards for messages so disparate systems could interoperate; we apply that lesson with Verifiable Credentials[6][7].

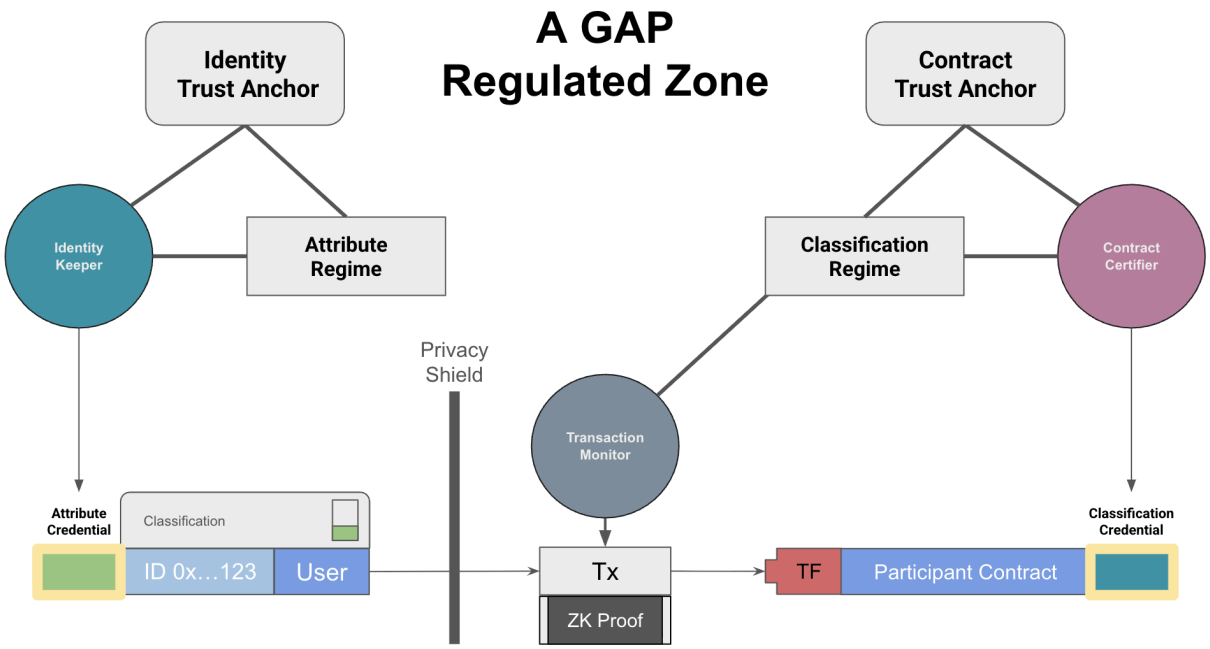

Figure 1: Visual overview of a Regulated Zone under the Global Access Protocol

Overlapping Regulated Zones allows for seamless composability across jurisdictions. Moreover, users and developers can be largely ignorant about the Regulated Zones themselves. Since service providers can operate in multiple Regulated Zones and under multiple Trust Anchors, users and developers can go to a single Identity Keeper and Contract Certifier, respectively, and receive many Verifiable Credentials simultaneously, granting them a great deal of access.

Although GAP is not a privacy protocol, it is compatible with privacy. Specifically

---

[3] Ethereum.org, "ERC-20 Token Standard" — "allows developers to build token applications that are **interoperable** with other products and services."
[4] EIP-721, "Non-Fungible Token Standard" — "allows for the implementation of a **standard API** for NFTs within smart contracts."
[5] Schär, *Decentralized Finance: On Blockchain- and Smart Contract-Based Financial Markets*, Fed. Reserve Bank of St. Louis Review — highlights DeFi's **composability** ("allows anyone to combine multiple applications").
[6] FIX Trading Community, "What is FIX?" — FIX is the **open standard** and "language of the global financial markets" used by thousands of firms daily.
[7] ISO 20022 (official), "About ISO 20022" — a common platform/standard for **financial messaging** that promotes interoperability across institutions and infrastructures.

- Users do not reveal personal information to transaction filters onchain, instead using ZK proofs
- No actor needs to have transaction data and personnel information. This prevents honeypots of data that are vulnerable to abuse or theft
- Since Transaction Monitors only monitor the traffic for a single contract, no single actor needs to aggregate this data, again preventing honeypots of data that are vulnerable to abuse or theft
- Because of the previous points, GAP can be deployed on a privacy chain like Aztec or Aleo without granting any single actor access to the entire shielded state
- GAP does not link any two transactions, allowing GAP to be composable with privacy tools that obscure the relationships between transactions.

3. Users and Attributes

3.1 Objectives

Many regulations set requirements for who can invest in certain assets. These regulations can be prohibitions, for example a sanctions list, or obligations such as recording PII. In traditional finance, such regulations are enforced through a Customer Identification Program (CIP) that intermediaries use to onboard customers before they grant access to their system.

An essential requirement for GAP is to enable the same types of controls to be placed on a Regulated Zone. However, as a decentralized protocol, it cannot rely on centralized administrators to white list users that will fragment liquidity and prevent smart contract compositions. Instead, GAP allows a Regulated Zone enables *Verifiable Credentials,* an established tool that allows one entity to digitally certify something about another entity.[8] Verifiable Credentials in GAP abide by specific standards which provide a common language for contracts to reason about users.

The credentialing mechanism in GAP achieves the following:

- Users do not reveal their PII on the ledger
- PII can be encoded into obfuscated onchain objects
- Standards for encoding PII can be shared amongst many participants
- Administers can enforce customer onboarding standards
- Any entity can create and enforce standards

[8] We note that DIDs is a technology well established outside of GAP that has many potential uses. https://www.w3.org/TR/did-1.1/

- One stop onboarding: a go through a single onboarding process to access a wide variety of regulatory zones

3.2 Digital Identities

Utilizing Verifiable Credential technology, users are represented in GAP by *Digital Identities (DIDs)* which are unique identifiers used to other modules to fetch and verify user information. A DID is useful because it is more permanent than a wallet address. Indeed a user can rotate their wallet addresses or use several simultaneous addresses for privacy reasons. However, a DID remains static.

Specifically, GAP will maintain a singleton contract called a *DID registry* that maintains a key value store, mapping user DIDs to *DID documents* where public information about the user is stored. When a user creates a new entry, the contract will create a unique DID using the transaction hash. Anyone can create a new DID, however the smart contract will charge a fee to discourage abuse.

A DID document will contain several fields, some mutable fields and some immutable. The most important mutable field will be a governance key that grants the user the ability to change the other mutable fields in their identity document. Another example of a mutable field is the *Spending Key Commitment*: the hash of a list of addresses holding user funds. An immutable field will be a commitment to the user's Private Salt, which is a critical privacy component discussed later in Pseudonyms and the Sybil Problem.

We stress that DID documents are public, and thus should only contain the hash sensitive information. The DID Registry will maintain a sparse Merkle tree of this key value store whose leaves are indexed by DID Documents, indexed by DIDs. We call the root of this Merkle tree the *DID Root*. This DID root will serve as a public input to ZK Proofs, allowing users to prove facts about sensitive data. For example, a user can prove that an address is in the Spending Key Commitment associated to some DID without revealing that DID. This allows a user to obfuscate the relationship between their DID and their funds.

3.3 Attribute Regimes and Attribute Credentials

An *Attribute Regime* is a data standard that specifies the type of an Attribute, an object encoding of user PII. As a simple example, we can define an "SDN List" Attribute Regime, that consists of a single field called "Sanction Status" which is a Boolean.

GAP will have an *Attribute Regime Registry*, a singleton contract where any party can define new Attribute Regimes. This singleton contract will be protected by a fee mechanism to prevent abuse. Each new Attribute Regime will be assigned a unique identifier for reference. For simplicity Identity Attributes will be immutable. Documentation for Attribute Regimes will be maintained on a neutral platform, similar to how ERC standards for Ethereum dApps are maintained via the

Ethereum Improvement Process (EIP)[9]. This open process will allow different Regulated Zones to share the same Attribute Regimes, promoting common standards across the space.

In the GAP protocol, a DID can be issued what is called *Attribute Credential* which is simply the hash of an Attribute. We say Attribute credentials are stored onchain, which is why it is important that they do not reveal the PII contained in the attribute. With an Attribute Credential, a user can prove to a smart contract that they have a certain attribute. This can either be done by revealing the attribute, or with a ZK proof.

In short, an Attribute Credential is an attestation that a user has a particular Attribute. But who is authorized to perform this attestation and how is it recorded?

3.4 Identity Trust Anchors and Identity Keepers

Attribute Credentials will be managed in an *Identity Trust Anchor Contract*, a factory contract that can be deployed via GAP. The purpose of this contract is to provide merkle trees that allow users to create a ZK proof that their DIDs hold valid Attribute Credentials. This contract is administered by an *Identity Trust Anchor*, who dictates how credentials are issued and who can issue them.

Specifically, the Identity Trust Anchor will manage which Attributes Regimes are supported in their contract, corresponding to their real world standards and policies. For each supported Attribute Regime, the contract will maintain a key value store, mapping DIDs to Attribute Credentials under the corresponding Attribute Regime. Furthermore, the Identity Trust Anchor maintains a sparse Merkle tree of this pairing whose root will be called the *Attribute Regime Root* that users can include as a public input of a ZK proof.

For each Attribute Regime, the Identity Trust Anchor will whitelist actors call *Identity Keepers* to issue and revoke Attribute Credentials. These actors will onboard users and store Attributes, since only hashes of Attributes are available onchain. They will also monitor the appropriate channels to know when credentials must be revoked, for example if a user is added to the SDN list. Moreover, Identity Keepers can operate on the multiple Identity Anchor Contracts[10]. This enables one step onboarding: users can onboard with a single Identity Keeper simultaneously receive a multitude of credentials simultaneously allowing them access to a variety of Regulated Zones.

GAP only allows each user to receive only one Attribute Credential under each Attribute Regime on each Identity Trust Anchor Contract. This will allow Attribute Credentials to serve as a Sybil protection mechanism, as discussed later in Pseudonyms and the Sybil Problem. Because the topic is rather technical, we refer the reader to Appendix III for further information.

[9] https://eips.ethereum.org/

[10] GAP will provide a singleton contract where Identity Keepers can register static identifiers and manage their keys.

As the gate keepers of a Regulated Zone, the Identity Keepers play an incredibly important role. The veracity of the Attributes in each credential depends entirely on the integrity of the processes of the Identity Keeper. Identity Trust Anchors have the power to eject Identity Keepers from their contract and revoke their credentials. Thus Identity Trust Anchors have the ability to set and enforce standards and CIPs (i.e. onboarding procedures) used to issue Attribute Credentials.

Other actors can white list the *Identity Trust Anchor ID*, namely address of the Identity Trust Anchor Contract, based on their reputation of their standards. Thus the pair, Attribute Regime and Identity Trust Anchor ID, form what is called an ontological standard: the pair not only standardizes the data type but the data collection methodology. An ontological standard is a glue that binds the digital world with the physical world.

Identifying different Identity Trust Anchors is also important because the Global Access Protocol does not enshrine a particular Identity Trust Anchor. Instead it supports a robust ecosystem with many different Identity Trust Anchors. Any organization can declare itself an Identity Trust Anchor and join the protocol, including an industry association, a self-regulatory organization, or some other standards body. In fact, the protocol will allow a new trust anchor to duplicate the credentials issued by another trust anchor. This provides an important check on the power of a trust anchor: if a trust anchor becomes corrupted or ineffective, another organization can simply fork it in the same way that a blockchain can be forked. Such a system can foster diverse innovation and prevent the protocol from stagnating and ossifying.

4. Smart Contracts and Classifications

4.1 Objectives

In order for a Regulated Zone to maintain its integrity, contracts must be able to control how they are composed with each other. Consider a token with strict controls that requires the holder to have a “Non Sanctioned” Attribute Credential. If this token is deposited into an ERC-20 contract or something similar, these controls no longer apply to the wrapper. Thus proper sanctions controls on a token, requires the token to be only deposited in contracts that also contain these controls.

To enable contract composition controls, GAP has the ability to credential contracts according to standardized classification systems. These classification systems allow rules and regulations to be enforced on a regulated zone: noncompliant contracts will simply lose their compliant classification. Furthermore, abiding by a classification allows the developer to abide by laws and achieve regulatory compliance.

Specifically, the classification system in GAP achieves the following:

- Contracts classifications are stored onchain
- Classification standards can be defined and shared across regulatory zones

- Administers can enforce customer onboarding standards
- Any entity can create and enforce standards
- One stop onboarding: a developer can receive many (and potentially) all classifications for multiple regulatory zones at the same time

There is an obvious parallelism between classifications and Attributes discussed in the previous section. However the machinery to enable contract classifications is much simpler. Attributes contain PII, and thus the protocol must allow users to prove the existence of these attributes via ZK proofs, requiring additional machinery such as anchor contracts which provide Merkle roots: discussed further later in Sessions. However, since contracts and their classification are public, such machinery is unnecessary.

4.2 Classification Regimes

GAP builds Regulated Zones using another set of ontological standards: a *Classification Regime* is a regulatory classification system of smart contracts. Specifically, a Classification Regime enumerates a set of *Contract Classes*, where each class specifies a set of requirements for the contract. The Classification Regime specifies for each Contract Classes both a human readable name and a binary code. Examples of Contract Classes can include "SEC compliant DEX", "Tokenized Security", and "AML/CFT screened token".

Like Identity Attributes Regimes, we envision Classification Regimes being proposed and maintained on an open platform similar to the EIP process. This platform will associate a binary identifier to every Classification Regime. This open system guarantees that a diverse set of interpretations of the law are available, and that compliance can evolve with legal thinking.

The exact nature of Class's requirements are outside the scope of the protocol, but they can include functional requirements of the contracts such as certain mandated function calls, security practices like code audits, or realworld legal obligations like filing an S1. An important example of a functional requirement would be a filtering mechanism (revisited later in Transaction Filters) that requires the caller to possess particular Attributes issued in Attribute Credentials. Similarly, a Contract Classes can require that the smart contract only composes specific Classess. Another important[11] A functional requirement is the absence of centralized points of control.

As an example, recall that we previously defined an "SDN List" Attribute Regime, that consists of a single field called "Sanction Status" which is either True or False. We now define a Classification Regime called "CFT" which consists of a single Class: "Sanctions Compliant." For a smart contract to be "Sanctions Compliant" it must require that the caller has an Attribute Credential under the "SDN Regime" with the "Sanction Status" Attribute set to false.

[11] Reddig, Rebecca; Mosier, Michael; and Gilman, Katja. "Genuine DeFi as Critical Infrastructure: A Conceptual protocol for Combating Illicit Finance Activity in Decentralized Finance," available at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4607332

GAP will indicate the classification of a contract via *Classification Credentials*, objects that contain the binary code for the Contract Classesand the identifier of the Classification Regime. These credentials can be used by wallets and other contracts to determine if a contract is a Regulated Zone and its function. Classification Credentials may also contain metadata, for instance Classification Credentials may indicate the *Contract Sponsor*, the party responsible for the contract. However the protocol does not dictate how this metadata is used.

4.3 Contract Trust Anchors and Contract Certifiers

We now describe how Classification Credentials can be issued in GAP and used to construct a Regulated Zone. The protocol provides a contract factory that allows actors called *Contract Trust Anchors* to deploy and administer their *Contract Trust Anchor Contract*. In this contract, the Contract Trust Anchor will maintain the list Classification Regimes they support and whitelist actors called *Contract Certifiers* that can issue Classification Credentials under each Classification Regime. The address of the Contract Trust Anchor will be the unique identifier for the Contract Trust Anchor.

GAP will maintain a singleton *Session Manager* contract where Contract Certifiers issue credentials. Specifically, the Session manager will maintain a key value pair store that maps an address to a *Contract Session*, an object containing the Classification Credential, the Contract Certifier, the Contract Trust Anchor, and other relevant information[12]. When a Contract Certifier starts a Contract Session, the Session Manager checks the Contract Trust Anchor Contract to see if the Contract Certifier has the appropriate permissions.

As with Identity Keepers and Identity Trust Anchors, Contract Certifiers will set standards and procedures for Contract Certifiers, to ensure that the Classification Regime is implemented correctly. By adopting a Classification Regime, the Contract Trust Anchor essentially defines their own Regulated Zone. They administer their Regulated Zone by setting rules of conduct and deputizing Contract Certifiers to enforce those rules via issuing and revoking Classification Credentials. A Contract Trust Anchor could potentially allow developers to register as Contract Certifiers and self certify, in which case they are directly overseeing the developers.

GAP is designed so that software developers abiding by a Classification Regime deploying contracts in a Regulated Zone can fulfill their legal obligations and minimize their liability. However, this depends upon the developer choosing Contract Trust Anchors (and Identity Trust Anchors) that administer robust Regulated Zones that comply with the relevant laws. Allowing anyone to deploy a Contract Trust Anchor Contract increases the decentralization of the protocol, but requires the participants to evaluate the Contract Trust Anchor's reputation.

[12] For example, GAP will define an interface that allows smart contracts to list versions of mutable smart contracts. The Contract Sessions will then list the version number, forcing developers to have each new version recertified. This prevents malicious developers from secretly introducing prohibited functionality such as back doors and exploits.

Lastly, we note that a single entity can simultaneously be both as an Identity Trust Anchor and as Contract Trust Anchor. One can imagine an entity needing to set both Attribute Regimes and Classification Regimes in a new jurisdiction without any Regulated Zones. However, the roles are separated in the protocol to introduce flexibility in the system. One can imagine different asset classes (e.g. tokenized commodities and tokenized securities) using Classification Regimes administered by different Contract Trust Anchor, but using a common Attribute Regime and Identity Trust Anchor. Alternatively, one can imagine that initially

5. Transaction Monitoring

5.1 Objectives

Attribute Credentials and Classification credentials are used to admit users and contracts into a Regulated Zone, and can be used to enforce a wide variety of regulations about who can transact and the types of platforms they can transact on. However many regulations pertain to how individuals can transact, including AML laws or Market Integrity rules. Such regulations require an entity to monitor transactions and apply a risk based approach to flag suspicious transaction patterns. In GAP, this is done by entities called *Transaction Monitors*.

Currently, blockchain analytic companies such as Chainalysis, TRM, and Elliptic monitor all public blockchain transactions, enabling regulators to police blockchains to a large extent. We refer to this activity as *Global Transaction Monitoring*.

Unfortunately, Global Transaction Monitoring is fundamentally at odds with privacy; it is impossible to have completely private transactions that can be monitored. One potential solution is to simply grant a Global Transaction Monitor the ability to monitor all private transactions in a Regulated Zone. However, this creates two problems. First, this creates a large honey pot of data that can be stolen or abused. Second, Global Transaction Monitoring is a large operation that requires explicit permission, and thus they will become entrenched actors conflicting with the open, competitive design principles GAP is built upon.

Instead GAP enables a different approach. It enables *Local Transaction Monitoring*: a Transaction Monitor can detect and flag[13] suspicious activity on a specific smart contract instead of at the global level. A Classification Regime will specify which classes of contracts require a Transaction Monitor and the types of behaviour patterns that must be flagged. Thus Local Transaction Monitoring thus enables a pragmatic approach to privacy, where data is shared on as needed basis to a diffuse group of actors.

[13] We note our word choice here. Often, flagged behavior will result in some sort of blacklist from the smart contract. However, flags can simply indicate an elevated level of risk that alone might not merit blacklisting.

Although GAP itself is not a privacy protocol, it prevents anyone from knowing both a user's identity and the transactions they are issuing, preventing vulnerable honey pots of data. Furthermore, GAP can be deployed on privacy chains like Aztec and Aleo. Lastly, since it does not directly link transactions, GAP can be composed with privacy pools and other privacy protocols, as discussed further in Appendix II.

Specifically, GAP enables the following:

1. Transaction Monitors can flag a variety of behaviour patterns while only observing the calls to a single smart contract
2. Transaction Monitors can determine when two transactions they are observing are issued by the same user
3. Transaction Monitors can refer a regulator to the Identity Keeper of a user for PII
4. Transaction Monitors cannot directly learn the DID associated to any user issuing a transaction
5. Besides Transaction Monitors, no other actor (including Identity Keepers) can directly tell if two transactions were issued by the same user.

The key is solving a fundamental problem in automated compliance: the *Sybil Problem* is when an attacker can create addresses ad nauseam and can transfer funds to these new addresses. The Sybil Problem creates two issues for Transaction Monitoring. First, sophisticated tracking is required to distinguish between actual transfers and obfuscated control. Second, it is impossible to create a canonical identifier for an actor that can be used to flag them or blacklist them from a smart contract: an attacker can bypass an address-based blacklist by simply creating a new address. Thus, effective address-based blacklists require a time delay for offchain analysis. This time delay breaks the inherent composability of DeFi.

5.2 Pseudonyms and the Sybil problem

The Global Access Protocol leverages Attribute Credentials to solve the Sybil problem. Recall that the protocol only allows each user to receive at Attribute Credential of each type with each Identity Trust Anchor. Thus a credentialed DID is a Sybil protected identifier.[14]

Unfortunately, including a DID in each transaction does not offer sufficient privacy protection, because a user's DID will often be linkable to their true identity. Recall that a DID is not PII: it is listed publicly in a digital identity smart contract. DIDs are frequently used in public applications, and thus a user's DID and real identity will likely be frequently linked in daily interaction. Eventually, PII, like a user's name, can be publicly associated with the DID, particularly if the user is a large organization. This is a variation of the "Doxing Slow Burn" discussed in the introduction.

[14] As remarked in Identity Trust Anchors and Identity Keepers and Appendix III, this is why it is important that every user has at most one Attribute Credential under each Attribute Regime issued under an Identity Trust Anchor.

To address privacy, we introduce *Pseudonyms*: these are Sybil protected identifiers derived from a DID, an Attribute Regime, and a smart contract address that does not leak the DID. We now walk through how a Pseudonym is generated. First, every user will store the hash of a secret random value called a *Private Salt* as an immutable field in their DID Document. The Private Salt thus is bound to the DID. From value, a user can generate their Pseudonym:

$$\text{Pseudonym} = \text{hash}(\text{User DID}, \text{Private Salt}, \text{Attribute Regime}, \text{Smart Contract Address}))$$

This pseudonym has the following properties:

- Each Pseudonym is unique given a user, Attribute Credential, and smart contract address
- Pseudonyms are Sybil protected with amongst users with the appropriate Attribute Regime
- Bound by the Private Salt, a user cannot choose their Pseudonym
- The DID is obscured by the hash

A user will use a User Session (discussed in Sessions) to link each transaction to their Pseudonym generated from the requisite Attribute Regime. However, we do not want the general public to link transactions via Pseudonyms, and thus we introduce *Encrypted Pseudonyms*:

$$\text{Encrypted Pseudonym} = \text{Encrypt}_K(\text{Pseudonym}, \text{Identity Keeper}, \text{Random Nonce})$$

With an Encrypted Pseudonym linked to each transaction, a Transaction Monitor can discover the Identity Keeper and the Pseudonym without revealing this information to the general public. Note a ZK proof is required to prove that an Encrypted Pseudonym is generated correctly, since the VM cannot distinguish between a genuine ciphertext and random bytes.

5.3 Onchain Risk Databases

The *Onchain Risk Database* is a smart contract enabling a Transaction Monitor to flag Pseudonyms. Specifically, each Transaction Monitor will use a contract factory to deploy their own Onchain Risk Database. This contract will maintain a key value store mapping hashed pseudonyms to a *Risk Flag*, which will simply be a fixed length byte string. Furthermore, the protocol will maintain a Merkle tree of this key value store whose root is called the *Risk Root*. The contract address for the Onchain Risk Database will be called the *Onchain Risk Database ID*.

A Classification Regime can require that certain smart contracts have a Transaction Monitor, and can specify objects for that. However, the exact behaviours that a Transaction Monitor must flag is outside the scope of the protocol, and is subject to policies of the Contract Trust Anchor. Similarly, the Contract Trust Anchor determines how these behaviours are encoded in the Risk Flag. Such a flexible and dynamic system is necessary since the behaviors and the exact transaction patterns will constantly evolve. Furthermore, Transaction Monitors will often want to keep this information secret, to avoid revealing their methods to criminals.

For example, in order to achieve a regulatory objective like AML screening, the Contract Trust Anchor might define the Risk Flags 0010 and 0001 as “Moderate Risk Structuring” and “High Risk Structuring” respectively, where structuring is a specific money laundering technique. The Transaction Monitor then can define a list of very specific patterns that suggest structuring. However, this information would all be kept secret so that criminals would not modify their structuring techniques to avoid detection.

5.4 Regulatory Reporting

In traditional finance, many intermediaries are required to report suspicious transactions. This includes Suspicious Activity Reports (SARs) mandated by the Bank Secrecy Act to combat money laundering or Consolidated Audit Trail (CAT) mandated by the SEC to promote market integrity. In this section, we discuss how GAP can be used to implement similar reporting mechanisms.

Specifically, a Classification Regime can mandate that a Transaction Monitor reports suspicious activity to a “Regulator” which can be a government agency, a Contract Trust Anchor, or some other entity. In this report, along with the relevant transaction data, the Transaction Monitor can list the Identity Keeper, the pseudonym and the Attribute Regime and contract address used to generate the pseudonym.

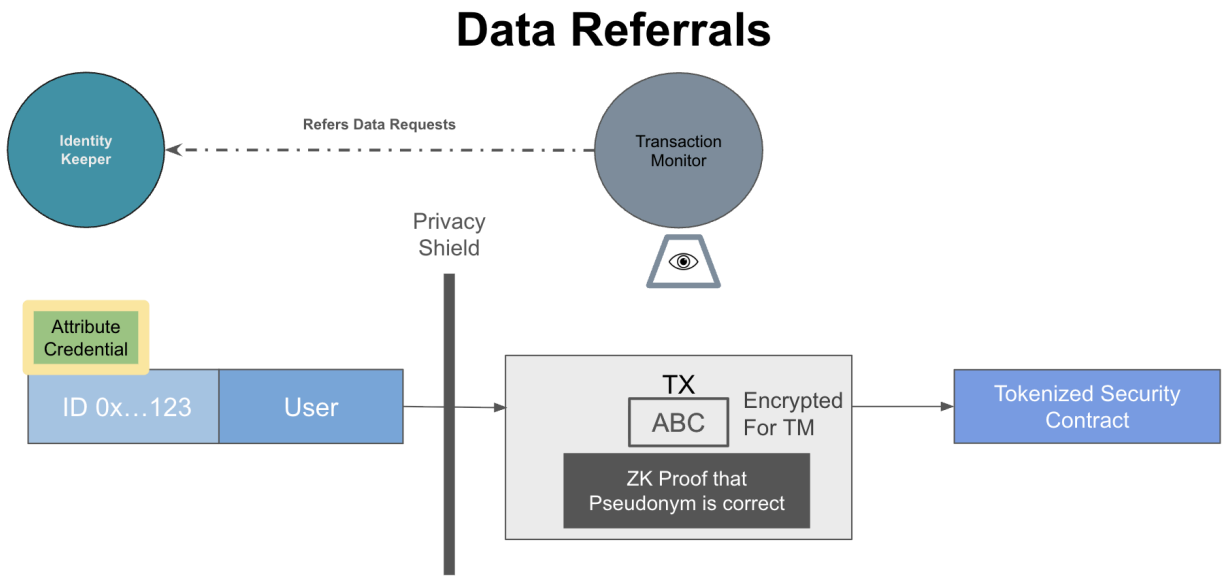

Figure 2: Although a Transaction Monitors does not have access to PII, they can refer any PII request to the relevant Identity Keeper

In traditional reporting schemes, these reports typically contain the PII of the individuals involved in the transaction. However, GAP specifically prevents the Transaction Monitors from having this information. Instead, the regulator can go to the Identity Keeper to fetch the necessary PII. Indeed, an Identity Keeper can require its users to reveal to them their Private Salt. In this case, given a

Pseudonym, Attribute Regime, and contract address, they discover the underlying DID by computing all the possible Pseudonyms of their users. They can give the regulator this DID, so they can create a pseudonymous database, or they can reveal the user's PII.

Note that these mechanisms are not part of GAP itself. The protocol itself does not specify how this interaction takes place, nor under what circumstances the Identity Keeper should comply with such a request.

6. Transaction Filters

6.1 Objectives

In order to enforce any type of rules, contracts must be able to control who can call them and how they are composed. GAP enables these controls through a module called a *Transaction Filter*. This mechanism essentially halts the execution of a transaction if the arguments do not have the appropriate permissions. The machinery we have previously discussed is designed so that the Transaction Filter can make such determinations that are required by various regulations. We work through the specifics of the Transaction Filter at the end of this section in Transaction Filter.

We note that GAP allows arbitrary contracts to have a filtering mechanism, as opposed to other protocols such as ERC 3643, which only imposes rules on token contracts. To see why this is necessary, consider for instance a DEX for swapping tokenized securities. The tokenized securities will of course need filters to implement a sanctions enforcement program. Although these controls are passed to the DEX (if you cannot move the assets, you cannot perform the swap), the DEX may need to enforce additional market integrity rules and filter out market manipulators. Thus the DEX needs to have its own Transaction Filter.

Specifically, the Transaction Filter enables the following:

- Transaction Filters can be implemented into arbitrary smart contracts.
- Transaction Filters can screen users out based on logical combinations of Attribute Credentials and Risk Flags
- Transaction Filters and screen contract composition by Class
- Access to a smart contract can be revoked in real time
- No transactions are not directly linkable to any Attribute Credentials, Risk Flags or DIDs
- GAP Transaction Filters do not require changes in standard smart contract interfaces
- The Transaction Filter is easier for a developer to incorporate to manage and to incorporate into contracts

Zero Knowledge Proofs allow a user to dissociate their transaction from their Attribute Credentials and DIDs. However, we cannot assume the contract can pass Transaction Filter the ZK proof since common contract interfaces have no mechanism to accept such a proof.

6.2 User Policies

A *User Policy* is a predicate function, accepting as inputs Attributes and Risk Flags and returning a Boolean. As we will see in Transaction Filter, User Policies essentially allow a contract to reject transactions from certain users. A User Policy must be more complex than simply checking the value of certain fields and flags (e.g. "Nationality" = USA AND AML Risk Flag = False). GAP allows User Policies to be constructed using a moderately expressive language[15], since robust risk based compliance programs require semantics to filter out complex risk categories. For example, suppose regulators learn that Syrian Nationals are using a specific type of structuring pattern to evade sanctions. Then GAP must enable a contract to reject users that are Syrians AND have a structuring Risk Flag.

This example shows that User Policies must be dynamic. Criminals are constantly adjusting their techniques, and compliance teams must constantly adjust their programs in response. To provide the requisite dynamism, User Policies will be published onchain in a *User Policy Registry* contract. Each User Policy will have a unique identifier called a *Policy ID*, and will list an administrator who can update the policy as needed. Thus User Policies can be updated at a moment's notice by a developer, regulator, Contract Trust Anchor, or whatever administrator is tasked with developing policies.

6.3 Sessions

GAP will maintain a singleton contract called a *Session Manager* that records the permissions, acting as a middle layer between the Transaction Filter and the user. We have seen the Session Manager before in Section REF, where it was used to issue Classification Credentials via Contract Sessions. In this section, we discuss a second type of session recorded by the Session Manager, *User Sessions* which indicates that an address is associated with a user that satisfies certain User Policies.

Specifically, to start a User Session, a user submits to the Session Manager their address, the ZK proof, relevant Encrypted Pseudonyms, and relevant Identity Trust Anchors. The ZK proof verifies many claims:

- The address is contained in the Spending Key Commitment of the user's DID
- There exist Attributes contained in a valid Attribute Credential issued to the user's DID
- The encrypted Pseudonyms are generated from the user's DID correctly
- Any relevant Risk Flags for these Pseudonyms

[15] To reduce the circuit size, initial versions of GAP may restrict this language to simply Boolean operators.

- The User Policy applied to the Attributes and Risk Flags returns True

The ZK proof must also contain the relevant Attribute Regime Roots and Risk Roots as public witnesses. Once the ZK proof is verified by the Session Manager, the address is mapped to the Policy IDs.

The User Session lists the Onchain Risk Database IDs and Identity Trust Anchors, so that the Transaction Filter can determine if the User Policy was evaluated according to the correct ontological standard. Lastly also the User Session lists the Encrypted Pseudonyms, so that they are available to the transaction monitor.

User Sessions will necessarily have an expiry time determined by the User Policy. While a User Session is active, it cannot be revoked. Indeed, since the ZK proof obscures the link between the session and the DID's Attribute Credentials the Pseudonym's Risk Flags, it is impossible for Identity Keepers to determine if a session needs to be revoked. Therefore, to enable real time revocation and flagging User Sessions must only be one block long. However, certain use cases like high frequency trading, a User Session could be extended longer. Long User Sessions can also enable non-interactive receiving addresses.

6.4 Transaction Filter

With heavy machinery of the Session Manager in place, the Transaction Filter is a rather simple mechanism indicating if an address has the appropriate sessions (either Contract Sessions or User Sessions) to serve as an argument of a function. Specifically, the Transaction Filter accepts a function, an argument type, and an address and then returns a Boolean. GAP provides a factory contract for developers to deploy Transaction Filters, listing the sessions that allow access to each desired function. A smart contract in a Regulated Zone can then call Transaction Filter whenever a sensitive function is called.

When specifying in the Transaction Filter the accepted Contract Sessions, developer chooses:

- The necessary Classes
- The accepted Classification Regimes
- The accepted Contract Trust Anchors

While specifying the User Session, the developers choose:

- The necessary User Policies
- The necessary Attribute Regimes
- The accepted Identity Trust Anchors
- The necessary Onchain Risk Databases

How these choices are made are outside of the protocol, however the developer potentially has a large amount of latitude. However, in order for the contract to be certified under a Classification Regime, the developer must make their choices in line with the rules and objectives of the Contract Trust Anchor.

7. An Example

To conclude this document, we illustrate how GAP can be used to regulate a swap between two tokenized securities, Token A and Token B: see Figure 7. This situation involves three smart contracts:

1. Token A contract
2. AMM contract
3. Token B contract

In our example, suppose that the Contract Trust Anchor has defined an SEC Classification Regime, defining a Regulated Zone that complies with securities law. Thus, this hypothetical regime would specify two classes of contracts: the tokens would have "Tokenized Securities" Classification Credentials and the AMM a "Compliant DEX" Classification Credential.

Under this hypothetical SEC Classification Regime, each of these three smart contracts would have Transaction Monitors and Transaction Filters. Transaction Monitors for the tokens would do AML/CFT screening, trying to detect money laundering and sanction evasion, while the Transaction Monitor for the AMM would be performing market integrity screening, scanning for behaviours such as wash trading or sandwich attacks. As can be seen in Figure 7, each Transaction Monitor has an associated Onchain Risk Database, and the transaction performing the swap has three Encrypted Pseudonyms, one associated with each Onchain Risk Database.

When preparing the transaction, the user's wallet can discover that three User Sessions are necessary to complete this transaction: two for the token contracts and one for the AMM. The wallet would also discover the User Policy, Attribute Credentials, Onchain Risk Databases and other information necessary to start the session. After ascertaining whether the user is eligible to call each smart contract, the wallet[16] can prepare the ZK proof to start the sessions.

The wallet next constructs a transaction that first calls the Session Manager before executing the swap. When that transaction is executed, the Session Manager verifies the ZK proof and starts the User Sessions. The AMM's Transaction Filter checks that the caller has the necessary User Session for the swap function. The AMM then performs two token transfers: in each token transfer the Transaction Filter checks that the "to" and "from" addresses have the appropriate session. The user has already started their User Session, and the AMM already has an active Contract Session

[16] We say wallet, but the exact software is outside the protocol. A browser extension or a front end could also prepare the ZK proof.

via their "Compliant DEX" Classification Credential. Lastly, the compliance of a transaction is provable if ever subject to further regulatory inquiry.

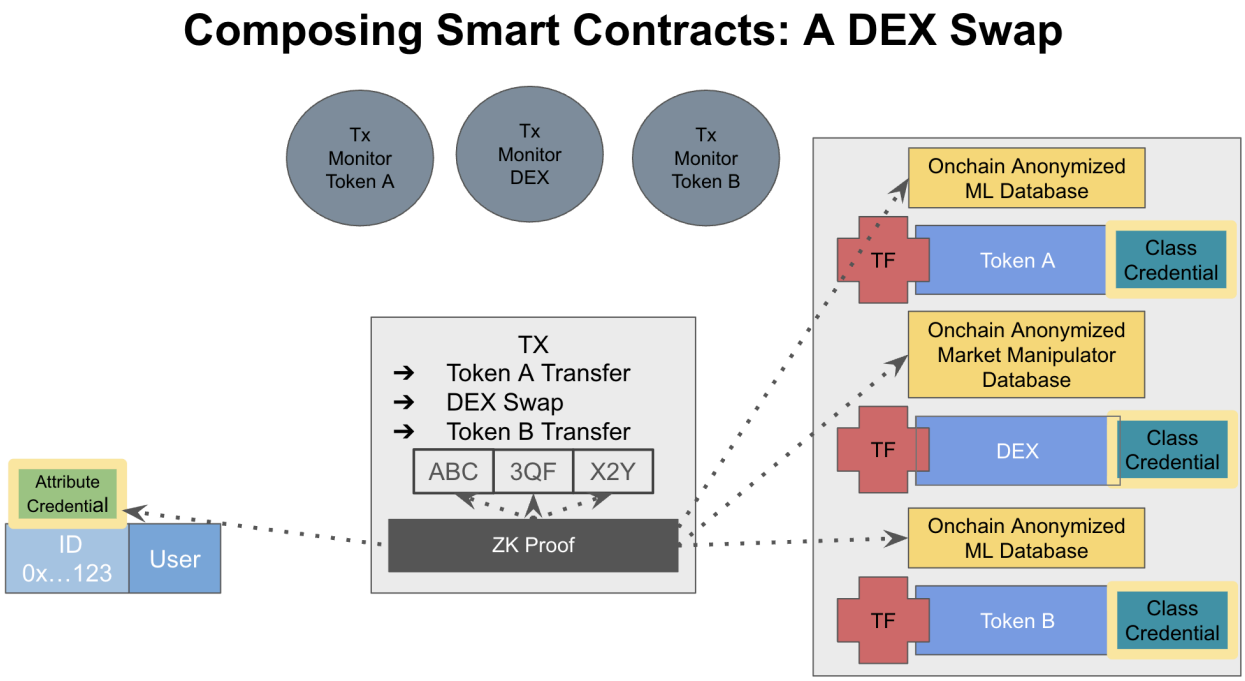

Figure 3: A DEX swap involves three smart contracts, each with its own Transaction Filter, Transaction Monitor, Onchain Risk Database, and Classification Credentials. The ZK proof verifies that the user has a permissible combination of Attributes and Risk Flags and has correctly generated the Encrypted Pseudonyms. Each Transaction Filter verifies th

8. Conclusion

As demonstrated, GAP is a protocol that allows a Contract Trust Anchor in coordination with an Identity Trust Anchor to deploy a Regulated Zone where participants, both software publishers and users, must conform to certain rules via class of Service Providers. The protocol provides a variety of features to allow the Attribute and Contract Trust Anchors to define the necessary standards and enforce the necessary rules.

Although we have tried to motivate each feature through discussion and examples, we have not illustrated in detail the value proposition of the Regulated Zones that GAP can be used to create. To that end, although they are outside the protocol, we outline several potential Regulated Zones.

AML/CTF Zone - This zone would enforce sanctions and prevent money laundering. Specifically, contracts in this Regulated Zone would have Transaction Filters that would screen out sanctioned users, and tokens would be required to have AMLTransaction Monitors. It would guarantee all assets in this zone would be clean.

Why would contracts opt into this Classification Regime? No one would want such a Regulated Zone to swallow DeFi into its Regulatory Model, nor do they believe that software developers should be subject to the BSA or sanctions laws.

First, various assets, including US securities, require effective compliance programs in order to be tokenized. Second, this Regulated Zone enables privacy friendly compliance. Compliant Centralized Exchanges must have compliance programs to ensure that they only allow clean tokens to be liquidated. Currently, as their only recourse, these compliance programs trace the history of the funds deposited on their platform. This implies that the transactions must be monitored by a blockchain analytics service, precluding privacy. However, an AML/CFT Regulated Zone offers an alternative compliance program.

Tokenized Securities Zones - In order to insure the integrity of US capital markets, US regulators enforce various rules on the issuance and trading of securities. These rules are designed to protect investors from fraud, information asymmetries, and exploitation. The tokenization of these securities presents a fundamental regulatory challenge.

GAP is designed specifically to create a Tokenized Securities Regulated Zone that resolves these regulatory challenges. First, issuance requirements such as disclosure rules for public and private securities can be enforced through Classification Credentials, with Transaction Filters enforcing investor status rules for non-public securities. Transaction Monitors for DEXs and other DeFi protocols can also protect markets from manipulation such as MEV attacks or Washtrading.

Moreover, the decentralized nature of Contract Trust Anchors can allow multiple interoperable Tokenized Security Zones. This will ensure competition and prevent a single Contract Trust Anchor or a single interpretation of the law to dominate the market and ossify.

This Regulatory Zone can also serve as a model for regulations of securities in foreign jurisdictions and for regulating other tokenized assets.

Anti-fraud Zone - This Regulated Zone can use Classification Credentials to indicate which contracts are not fraudulent. The Classification Anchor then can enforce ethical standards that protect users from predation. Wallets can restrict themselves to only interacting with contracts in the Anti-fraud Zone, protecting the user from scams. Such guardrails can be a very important feature for common retail who lack the experience to distinguish between legitimate platforms from fraudulent ones.

Anti-theft Zone - The vulnerability to theft is an unfortunate side effect of self custody: when coins are stolen in some hack or exploit, the recourse for regulators and owners is scant. The value can be traced, preventing the thieves from liquidating the tokens and legitimate exchanges. However, this seldom leads to recovering and the requisite work makes this impractical except large value thefts,

GAP can solve this problem with an Anti-theft Regulated Zone. The onchain on and off ramp to this Regulated Zone will be a special wrapper contract: only assets that are wrapped via this contract can enter the Anti-Theft Zone. To leave the Anti-theft Regulated Zone, the assets must be unwrapped, but the unwrapping function has a significant time delay. If token are stolen, the relevant Transaction Monitor can freeze the funds and broadcast Pseudonyms for wrapper Transaction Monitors to blacklist. Thus the funds can be prevented from leaving the Anti-Theft zone until they are located and frozen. Furthermore, the thieves can be prosecuted since the Identity Keepers have their PII.

Such a Regulated Zone would not be atomically composable with the rest of DeFi. However, this zone could be used to protect funds in cold storage.

Appendix I: Glossary

Attribute: A standardized data object encoding a specific piece of PII about a user.

Attribute Credential: An onchain attestation that a DID possesses a particular Attribute. It contains the hash of an Attribute and enables users to prove that they have that Attribute.

Attribute Regime: A data standard defining the type (i.e. structure) of Attributes.

Attribute Regime Registry: A singleton smart contract where Attribute Regimes are registered and assigned unique identifiers. It enables open standardization and reuse of Attribute definitions across Regulated Zones.

Attribute Regime Root: The Merkle root maintained by an Identity Trust Anchor Contract representing the current state of all issued Attribute Credentials under an Attribute Regime issued in that contract. It is used as a public input in ZK proofs.

Classification Credential: An onchain credential attesting that a smart contract belongs to a specific Contract Class under a defined Classification Regime.

Classification Regime: A standardized regulatory framework that defines a set of Contract Classes, each with specified technical or legal requirements. It serves as the basis for constructing a Regulated Zone.

Contract Trust Anchor: An entity that adopts Classification Regimes and dictates which Contract Certifiers can issue which Classification Credentials.

Contract Trust Anchor Contract: A smart contract deployed and administered by a Contract Trust Anchor to manage Contract Certifier permissions.

Contract Certifier: An authorized actor whitelisted by a Contract Trust Anchor to issue and revoke Classification Credentials for smart contracts under a given Classification Regime.

Contract Session: A record stored in the Session Manager linking a smart contract address to active Classification Credentials.

DID: A Decentralized Identifier representing a user within the protocol.

DID Document: An object controlled by the user storing public information pertaining to the DID. It can contain commitments to private data such as Spending Keys and Private Salt.

DID Registry: A singleton smart contract that maps DIDs to DID Documents and maintains the DID root.

DID Root: The Merkle root of the DID Registry. It serves as a public input for ZK proofs involving user identities.

Encrypted Pseudonym: A ciphertext derived from a Pseudonym using a Transaction Monitors key.

Global Transaction Monitoring: The practice of monitoring all blockchain transactions across all blockchain networks.

Identity Keeper: A service provider authorized by an Identity Trust Anchor to onboard users, collect PII, and issue Attribute Credentials.

Identity Trust Anchor: An entity that administers Attribute Credential issuance via the Identity Trust Anchor Contract. It adopts Attribute Regimes and dictates which Identity Keepers can issue which Attribute Credentials.

Identity Trust Anchor Contract: A smart contract through which Attribute Credentials are issued and provides Attribute Regime Roots. This contract is deployed and administered by an Identity Trust Anchor.

Local Transaction Monitoring: A model where a Transaction Monitor observes and analyzes activity for a specific smart contract rather than the entire blockchain.

Onchain Risk Database: A smart contract deployed by a Transaction Monitor to record Risk Flags.

Onchain Risk Database ID: The contract address of a specific Onchain Risk Database.

Policy ID: A unique identifier assigned to a User Policy in the User Policy Registry.

Private Salt: A secret random value committed in a DID Document for generating Pseudonyms.

Pseudonym: A Sybil-resistant identifier derived from a DID, Private Salt, Attribute Regime, and smart contract address. It uniquely identifies a user for a specific contract without revealing the DID.

Regulated Zone: A collection of smart contracts operating under common standards administered by a Contract Trust Anchor.

Risk Flag: A fixed-length byte string recorded in an Onchain Risk Database indicating that a Pseudonym has been associated with a specific risk condition.

Risk Root: The Merkle root of an Onchain Risk Database's Risk Flags, used as a public input in ZK proofs to verify risk status.

Session Manager: A singleton smart contract that records and verifies User Sessions and Contract Sessions, acting as an intermediary between users and Transaction Filters.

Spending Key Commitment: A hash stored in a DID Document representing the set of addresses controlled by a user, enabling them to dissociate their holdings from their DID.

Sybil Problem: The challenge of preventing a single actor from creating multiple identities to evade controls.

Transaction Filter: A smart contract module that determines whether a transaction may proceed by checking for valid User Sessions and Contract Sessions.

Transaction Monitors: Service providers responsible for observing contract activity, detecting suspicious behavior, and issuing Risk Flags in Onchain Risk Databases.

Trust Anchor: A general term for Identity Trust Anchors and Contract Trust Anchors.

User Policy: A predicate function that evaluates Attributes and Risk Flags to determine whether a user satisfies specific compliance requirements.

User Policy Registry: A smart contract that stores User Policies and manages governance.

User Session: A short-lived record in the Session Manager indicating that a user address satisfies specified User Policies and has provided the necessary ZK proofs.

Verifiable Credential: A cryptographically signed attestation issued by an authorized entity certifying specific claims about a user or contract.

Appendix II: Privacy Vaults

While powerful and compatible with GAP, privacy chains often lack the adoption of their public counterparts, and thus public blockchains currently have the deepest liquidity. Luckily the Global Access Protocol allows a novel[17] privacy tool for public blockchains: a *Privacy Vault* that allows users to obscure the connections between transactions. Similar to a mixer or a privacy pool, users deposit their funds into a pool where they "mix" with other tokens, and then the user withdraws them later, proving their ownership with a ZK proof. The ZK proof obscures the exact origin of the funds, using the other tokens in the pool as a screen. However, contrary to mixers, Privacy Vaults deployed in a regulated zone can require that the depositor and the withdrawer share the same DID.

Mixers are a privacy tool on public blockchains that allow a user to obscure the history of their funds from blockchain analytics services. Some mixers, including Tornado Cash, are used for money laundering, and thus invite regulatory scrutiny.[18] Furthermore because mixers effectively foil blockchain analytics, and funds that can be traced back to a mixer must be assumed to have high money laundering risks. As such, certain platforms such as centralized exchanges are forced to censure any such funds, rendering mixers unusable for typical users.

In contrast, suppose an asset with a Transaction Monitor is deposited into a Privacy Vault in some Regulate Zone. Since the token cannot change hands within the vault, the Transaction Monitor can still deduce the balance of each pseudonym, simply by monitoring the deposits and withdrawals. As such, the Transaction Monitor can perform effective AML screening on that token.

How much privacy do Privacy Vaults afford within the GAP protocol? When used correctly, the answer is quite a bit. Suppose that:

- A user keeps all their tokens stored in the Privacy Vault.
- Every time they transact, they pull funds out of the mixer, perform the desired operation, and then place the funds back into the mixer.
- Every transaction, the user uses a different wallet address.[19]

Following this strategy, there is nothing directly[20] connecting the different transactions together, since the actual balances are encrypted and a different wallet address is used. Thus by working on

---

[17] The authors would be remiss if they did not mention the following influential paper: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4563364

[18] See the founders' indictment.

[19] The astute reader will notice that we have omitted the problems associated with paying for gas. Gas requires funds to be stored at an EOA address, and the management of gas fees inevitably links transactions together. This problem can be solved with a gas sponsorship scheme.

[20] Unfortunately, some guesses can be made indirectly. If an observer sees 3781 tokens deposited into a mixer, and then 3781 withdrawn the next day, they can guess that it was the same user who performed both operations.

public infrastructure, Privacy Vaults offer a simpler, although less robust, privacy solution from privacy chains.

Appendix III: Sybil Resistant Credentials

To enable robust Transaction Monitoring and effective Pseudonyms, we must solve the Sybil Problem: we must guarantee that every user only receives one Attribute Credential under the auspices of each Identity Trust Anchor. Limited only by gas fees, a user can create as many DIDs via GAP. Thus, when a user onboards with an Identity Keeper, they must be able to determine if some other Identity Keeper has already issued an Attribute Credential to another DID controlled by that user.

Note that our system does not require a perfect Sybil Protection Mechanism. If a user is able to maintain a few duplicated credentialed DIDs, Transaction Monitors can still perform their duties. Furthermore, such duplications are not only detectable, but can constitute fraud, and thus the threat of prosecution can mitigate the Sybil Problem. Moreover, the more identities an attacker attempts to maintain, the more likely they would be to be caught.[21]

In this appendix, we explore two mechanistic solutions to the Sybil Problem that can be employed by the Trust Anchor. Although the solutions discussed here are not considered to be strictly part of GAP, the authors wish to record their work on the subject for the benefit of future developers wishing to implement infrastructure on top of GAP.

III.i Basic Solution

In GAP, every Attribute Credential has a metadata field. The Identity Trust Anchor can choose a specific identifier that is part of the User's PII and require an Identity Keeper to post the hash of that identifier in that Attribute Credential Metadata. When a user onboards with a new Identity Keeper, they can compute the hash of the identifier, and then check if any other credential issued under the auspices of the Identity Trust Anchor contains that hash.

This would effectively solve the Sybil Protection problem, assuming that all Identity Keepers collect the necessary identifier. However, this solution is susceptible to the following attack: a *rainbow attack* is when an attacker simply hashes all possible combinations of the unique identifier. For example, the hash of a 10 digit Social Security Number is susceptible to rainbow attacks: the attacker can easily precompute the hashes of all 10^11 possible social security numbers. Thus, if the hash of a Social Security Number is published in a credential, then an attacker can easily link the DID to the user's Social Security Number. Since a DID is not necessarily private, this effectively reveals an important piece of PII of the user.

[21] The authors note that the threat of prosecution may be a sufficient solution to this problem, since attackers probably cannot create credentialed DIDs ad nauseum.

An Identity Trust Anchor trust anchor can potentially choose a higher entropy identifier, such as the social security number concatenated with name and birthdate. However, high entropy PII, such as name, often have ambiguous data encodings. Is a person's name written as John Smith, Jonathan Smith, or John R. Smith? One solution is to build the identifiers from specific fields of a specific government issued ID. Then the identifier can be built from the official first, middle, and last name deterministically. But, then we have another problem: not all users will have the same government document, particularly if we consider non-US users. In this case, the Identity Trust Anchor will have to choose a plethora of accepted documents, and users can potentially create a credentialed DID for each document.

As discussed earlier, some duplicate credentials are acceptable. Thus, it is conceivable that an Identity Trust Anchor can find a suitable identifier. However, using a simple identifier like a social security number maybe simpler, in which case we need to improve our Sybil Protection solution.

III.ii Double Salt Solution

We now outline a more sophisticated solution, relying on elliptic curve cryptography. Before we begin, we set some notation.

- Let G be a group with generated g, and assume the Discrete log problem is hard with respect to g and G.
- Let A and B be two identity Keepers. In our solution, we assume that Identity Keeper A is issuing credentials, and Identity Keeper B is checking if a user has a credential
- Let p_U be an identifier of a user U, such as a Social Security Number. We do not assume p has enough entropy to resist rainbow attacks.
- Both Identity Keepers have access to a public database. This can be a blockchain or some message board.

We now outline the scheme.

1. The Identity Keepers and A and B generate secret hashes S_A and S_B respectively. We will call these their salts.
2. Both Identity Keepers A and B publish points q_A=(S_A)g and q_B=(S_B)g respectively.
3. When Identity Keeper A issues a user U an Attribute Credential C, they publish f_C=(p_U)(S_A)(q_B) on a public database along with the Attribute Regime and Identity Trust Anchor. If Identity Keeper A revokes the credential, they remove the relevant attribute from the public database.
4. When Identity Keeper B onboards a user U, they check if (p_U)(S_B)(q_A) equals f_C for any value on the public database. If so, the user U already has a credential issued by Identity Keeper A.

The last step of the scheme works, because if the user U has been onboarded, both f_C and (p_U)(S_B)(q_A) equal (p_U)(S_B)(S_A)g. Furthermore, since (S_A) and (S_B) are random hashes, they both act as salts giving both (p_U)(S_A) and (p_U)(S_B) sufficient entropy so that f_C is immune to rainbow attacks via multiplication by q_B or q_A respectively. In fact, as the holder of the secret value S_B, Identity Keeper B is the only entity that can meaningfully query if p_U is published on the database. Thus p_U can only be connected to a DID by Identity Keepers A or B.

Unfortunately, Identity Keeper B can potentially perform a rainbow attack: they can iterate through all values of p to see when p(S_B)(q_A). By examining when the entry was posted into the database, and comparing this with the credentials issued at the same time, Identity Keeper B can potentially connect DIDs, and hence users, to their PII. To solve this issue, the Identity Trust Anchor could maintain the database and rate limit access. Thus if Identity Keeper B attempts a rainbow attack, the Identity Keeper can bar them from access.

Unfortunately, this approach involves a fair amount of computation. In a system with n Identity Keepers and m credentials, the database would need nm, entries. Moreover, every time a new Identity Keeper is added, each existing identity Keeper must compute a new f_C for each new credential they have issued.