The Vega Upgrade: Data Portability and Transformations

Vana Core Developers
developers@vana.org

Abstract. Data sovereignty has moved from a niche concern to a mainstream requirement, driven by AI agents that transact over personal data at machine frequency, institutions adopting sovereignty as policy, and users who will never tolerate crypto-native complexity. The Vega upgrade makes data portability a first-class citizen of the Vana protocol, combining web2-level performance and UX with the sovereignty guarantees of the initial implementation. This addendum specifies the primitives that enable it: grants, scoped and revocable authorizations whose issuance, exercise, and revocation are distinct chain events and the protocol's unit of pricing; transformations, records computed from other records that carry lineage, allowing users to share precisely the transformation of their data a counterparty needs while preserving provenance even through deletion; the personal server as the transformation runtime, available in browser, device, and hosted variants with identical protocol-level guarantees; and protocol-governed encryption, under which no party, including the user's own personal server, can decrypt data without a valid onchain grant, enforced by a dedicated threshold quorum separate from consensus validators. End-to-end permissioning completes in roughly 50 milliseconds in production. A revised fee model meters reads as the event that scales with usage, and fee sponsorship lets applications absorb costs so users need never hold tokens. Aggregate grants extend the same semantics to data pools. Together, these changes make Vana the best network for sovereign, portable data, built on distributed infrastructure.

1. Motivation

Data sovereignty has grown from a niche market of passionate libertarians and decentralization advocates to something everyone interacting with AI thinks about. Companies wonder whether their proprietary data is being trained on, looking for solutions to lock down their core IP. Users wonder where their data actually goes, and, at the same time, want to pull it in to give AI the right context. People no longer see their data as exhaust from using the internet, as it is now a key ingredient, whether ChatGPT or Claude history, or call recordings, for making AI work well.

It's important to note that an interest in data portability is not an interest in a distributed network. Users, developers, and companies who want data portability and data sovereignty simply care

whether they can get the security and sovereignty guarantees they need in a functional form that is comparable to other APIs and centralized data infrastructure they are used to using.

Agents. An agent acting for a user reads and writes continuously, pulling context before a task, writing results back after, under a grant the user issued once. Every agent interaction is a portability transaction, executed at machine frequency by software that will never see the chain underneath. Agents need data transactions to be fast, seamless, and always have up-to-date data.

Institutions. From the PDPP standard, developed with the Linux Foundation, to governments seeking sovereign infrastructure, institutions are adopting data sovereignty as policy. What an enterprise is adopting is the assurance that its data remains its own; what a government is adopting is the assurance that its citizens retain control over their own data.

Users and builders. Participating in the network today requires crypto knowledge on both sides: users hold tokens and sign transactions; builders reason about sources, records, and keys. That was the right tradeoff for early adopters. Mainstream adoption requires a lower barrier.

Together these ask for something no network has offered: the performance and simplicity of a web2 platform, with sovereignty guarantees no web2 platform can make. The Vega upgrade is built for that intersection. It makes data portability a first-class citizen of the protocol: web2-level UX and performance, with the strong sovereignty guarantees of the initial implementation maintained.

2. Grants

A grant is a scoped, revocable, programmable authorization with five fields:

- **issuer:** the user
- **grantee:** an application, agent, or other party
- **scope:** which data and which operations
- **expiry:** when the authorization lapses
- **conditions:** predicates that must hold at exercise, such as fee payment or a region check

This is the object Section 2.1 of the whitepaper describes: access permissions as who can access what data, under what conditions. The grantee is the who, the scope the what, and expiry and conditions the circumstances. What this addendum adds is a fixed set of fields and a lifecycle: issuance, exercise, and revocation are three distinct chain events rather than a single state entry.

Protocol-level pricing attaches to the grant, its issuance and each read exercised against it, rather than to the underlying file. This is what makes the fee structure in Section 6 composable, and it removes any need for the protocol to be opinionated about file structure.

3. Transformations

A transformation is a record produced by running an operation over one or more source records. It is registered like any other record, with one addition: a lineage pointer back to its sources.

Operations are arbitrary compute over data, whether a redaction script, a summarizer, a topic categorizer, or a prediction model. The protocol does not prescribe which operations exist, only that their outputs are registered with lineage and that access can be granted over the outputs. Transformations form a tree over the raw data, each node carrying lineage to its parent, and the user grants access at whichever node matches what a counterparty actually needs. A user could grant access to the health-related information from their ChatGPT history, for example, without granting the history itself.

A transformation is either computed once and frozen (a snapshot) or recomputed as its sources grow (a view). The distinction determines what a grant is on: a snapshot grant covers a static artifact, while a view grant is a standing subscription to a result that tracks its sources. It also makes caching a property of the primitive rather than a separate layer, since a repeat request resolves to an existing node instead of re-running the operation.

There are no unregistered outputs. Any read that computes over data rather than projecting a stored scope produces a registered transformation, because without a registered node there is nothing to grant, nothing to revoke, and no lineage to audit.

3.1 Relation to cross-platform datasets

Section 3.3 of the original paper described normalization as something DLPs perform, enforcing structural consistency across sources through their validation mechanisms. That remains the case.

Transformations bring the same capability to the individual. A user can normalize, restructure, and summarize their own data independent of any pool, and grant the result directly. The cross-platform dataset described in Section 3.3, in which email, browsing history, and social activity are combined into a structured whole, is describable as a transformation tree over one user's records without depending on a user joining a DLP.

3.2 Provenance and lineage

Section 3.2 of the original paper specified privacy-preserving proof of contribution with two verification paths, zero-knowledge proofs generated locally and secure computation in the Satya Network, following a standardized attestation schema. Transformations build on this.

Where the original paper specified verification mechanisms, this addendum standardizes the metadata format that records attestation source and leaves consumers to filter on what they trust. A grant can carry an attestation from a signed export by the originating platform, a zero-knowledge proof over a TLS session, or an attestation from a specific operator the consumer already trusts. The way data is verified depends on each data source and how it comes in, so this

flexible format mirrors the actual structure of trust in production data pipelines, where verification is a chain of attested origins rather than a single proof.

For transformations, lineage is provenance. A consumer of a health summary can walk the tree to the export it was computed from and to that export's attestation. Provenance is maintained through these transformations.

3.3 Deletion

Deletion is the user's choice, exercised at the node. A user can delete a full lineage or a single node, and a deleted node's content is actually deleted. A transformation survives the deletion of its sources: a user can delete a raw export and keep the summary computed from it.

Deletion of content does not break provenance. Lineage retains the deleted source's hash and attestation, so a consumer walking the tree can still verify what a surviving transformation was computed from and how its origin was attested, without the deleted content being recoverable. Provenance is a property of the metadata, and the metadata outlives the ciphertext.

4. Personal servers as transformation runtime

Section 2.2 specified the personal server as the secure foundation for data sovereignty, whose core purpose is to run operations on a user's unencrypted data, and noted that it can be run by the user on their machine, by a trusted provider, or client-side in a lightweight way. That specification holds; this addendum names the operations.

The personal server is the transformation runtime. It executes authorized operations over the user's data and registers the output as a new record with lineage. It is pull-only, in that it never initiates outbound calls and only responds. It is an interface contract with multiple implementations.

Variant	Process location	Availability
Browser	The user's browser tab, as a lightweight user-controlled server	While the tab is active
Device or VM	The user's desktop or mobile device, or a VM the user provisions	While the device is active; always-on for a VM
Hosted	A provider-operated server running inside a trusted execution environment	Always-on

Protocol-level security guarantees are identical across the three: it is a trusted compute environment where the user chooses to work with their plaintext data. Depending on the compute requirements for the

operations a user needs to run, and the trust assumptions they would like if they choose to use a hosted provider, they can choose a personal server that fits their needs.

Section 4.2 established trusted execution environments as the setting for complex operations on private data, including arbitrary code for proof-of-contribution and model training. The hosted personal server applies that same guarantee to individual transformations, preserving the property that the requester never sees the inputs.

The personal server contract is open. Users, or third parties acting on their behalf, can run alternate implementations against it.

5. Protocol-governed encryption

For a grant to be more than a contractual assertion, decryption must be gated by the same state that records the grant. This is protocol-governed encryption: no party, including operators and the user's own personal server, can decrypt data without a valid grant onchain, ensuring a full audit trail of every operation or read on private data.

5.1 Node populations

The original paper specified validators securing consensus under Proof-of-Stake, and the Satya Network of trusted execution environments performing validation. This addendum distinguishes a third population.

Validators run consensus and secure the chain that records grants and settles fees. They are slashable for malicious behavior, and they have no access to user data.

PGE nodes form a threshold quorum that gates every decryption. No party, including the user's own personal server, can unwrap a file key without quorum participation, and each node independently verifies chain state before participating.

Data validators support aggregate operations inside a trusted execution environment. This is the role Section 3.2 assigned to the Satya Network.

Separating PGE nodes from validators is deliberate, because the threat models differ. A slashed validator should not, as a consequence of having once been a validator, retain the ability to decrypt data it once attested. PGE nodes therefore run a different binary, hold different key material, and carry a different operational footprint.

5.2 The read path

Per-file encryption uses a fresh symmetric key. Writing a file:

1. The SDK generates a random per-file key, 32 bytes, AES-GCM.
2. The file is encrypted with that key and the ciphertext is uploaded to storage.
3. The file key is encrypted under the PGE threshold scheme, such that its release requires quorum

participation. The wrapped key is anchored onchain.
4. A file reference, comprising location, hash, and scope, is registered on the chain.

Reading it, on a valid grant:

5. The personal server verifies the grant against the chain: grantee identity, scope, expiry, revocation status, fee record.
6. The personal server fetches the ciphertext for the requested scope from storage and the wrapped key from the chain.
7. The personal server requests key release from the PGE quorum. Each node independently verifies the personal server’s authorization, the same chain state, and, on threshold agreement, participates in releasing the key for this read. Released material is transient, so no standalone decryption capability accumulates at the personal server.
8. The personal server decrypts, projects the requested scope, and envelope-encrypts the projection to the grantee's public key.
9. The envelope is delivered over authenticated TLS and the grantee's SDK decrypts it with its own private key.

A transformation follows the same path but ends differently. Steps 5 through 7 run as for a read, authorized by a grant whose scope names the operation. At step 8 the personal server decrypts the sources and executes the operation instead of projecting a scope. The output is a new file: fresh key, encrypted and uploaded as in steps 1 through 3, registered onchain with a lineage pointer to its sources. The requester receives only the envelope of the output, never source plaintext, under a separate grant.

Plaintext only ever exists inside the personal server during a read or transformation, and inside the grantee's process after delivery. Everywhere else — in storage, on the wire, on the chain — data is ciphertext or metadata. The grantee never holds the file key, only an envelope wrapped to its own public key, so a file can be shared with a new grantee without re-encrypting it.

End-to-end permissioning latency, from grant issuance through verification, is on the order of 50 milliseconds in production today. The dominant costs in a read are not the cryptography but the chain round-trip for grant and fee verification and the quorum round-trip for key release. Transformation throughput is a property of the personal server variant executing the operation, not of the permissioning path.

5.3 Trust assumptions

Section 4.3 required trust in trusted execution environment hardware, the underlying cryptographic primitives, and an honest majority of validator stake. The read path above extends that list.

The protocol requires trust in the underlying cryptographic primitives, now including the threshold re-encryption scheme alongside ECIES, AES-GCM, and the chain's signature scheme; an honest majority of validator stake; an honest and live threshold of the PGE quorum, whose honesty bounds confidentiality

and whose liveness bounds availability, since it sits on the read path; and the personal server implementation the user is running, together with its host in the hosted variant.

The negative claims in Section 4.3 are preserved. The protocol does not require trust in any individual validator, any individual PGE node, any specific personal server host, any DLP operator, or any specific application developer. As stated there, it cannot protect against fundamental breaks in the underlying cryptography, physical compromise of user devices, or social engineering of users into surrendering their keys.

6. Fees and network sustainability

Section 5.3 of the original paper enumerated fee-bearing operations across data, token, and market categories. This addendum specifies protocol-level pricing for data portability events and directs where fees flow.

Writes are a fee-bearing operation. No rate is currently set, so registering data to the network is free today. The rate is a parameter and can be set to a non-zero value as the network grows.

Grant issuance carries a rate of zero today, so authorizing a counterparty is free. Like every other rate, it is adjustable.

Reads are the fee that matters: every access executed against a live grant, whether of a raw record or of a transformation, carries a read fee. A single rate applies today, and the architecture allows rates to vary by scope as usage patterns develop.

Read fees scale with how deeply the network is used. Section 5.3 listed the granting of access rights as a fee-bearing operation but did not identify the read as the metered event, and the read is where volume accumulates once data is in continuous use. Metering it is what makes the fee pool grow with usage rather than with onboarding alone.

Fees are payable in the native token or in whitelisted currencies such as USDC.

The protocol also supports fee sponsorship. A builder or gateway can hold a pre-funded balance from which fees are drawn when a user-triggered event fires, so the economic payer can be the application rather than the end user. An application can absorb its users' data access costs the way a web platform absorbs bandwidth, and consumers can experience the product without holding tokens or paying per read. This closes a practical gap in Section 6.3, where users simply logging in with their Vana identity would otherwise imply a token balance.

Fees fund the network's security and growth. Token holders can choose to stake on an operator, ensuring that security scales as the network scales. A portion of fees is directed to the ecosystem fund, which during the bootstrapping phase funded DLP rewards and is allocated to programs that underwrite growth of the network, bringing more data and users in. A portion of fees supports a burn mechanism, specified in the economics paper.

Supply and the role of VANA are otherwise as specified in Section 5 of the whitepaper.

7. Aggregate grants

The argument in Section 3.1 stands: a researcher or an enterprise cannot practically negotiate thousands of separate access agreements, and much as labor unions aggregate worker bargaining power, aggregations of data aggregate data bargaining power. An aggregate grant authorizes access to an aggregation as a whole, rather than to each contributor individually. It flows through the same rails as an individual grant, with one difference: it executes in a data validator rather than a personal server, since no single user's personal server is the right trust boundary for aggregate data.

DLPs use the same grant semantics, including expiry, conditions, instant revocation, chain-level audit, as coordination units built on the protocol. Aggregate access relies on trusting a DLP's proof of contribution and other aggregate requirements: that trust sits at the application layer, scoped to consumers who choose to exercise aggregate grants.

8. Conclusion

Data portability has existed as a legal right for years, and as a practical reality for early adopters. Mainstream and institutional adoption set a much higher bar for performance and UX. The Vega upgrade ensures this bar is met for applications and agents that are increasingly differentiated by the right access to context, making portable data critical. Vana is the best network for sovereign, portable data, built on distributed infrastructure.