

The Discovery of the Consent Constant

A Scientific Examination of $\kappa(c)$ and Cross-Network Authorization Invariance

Mats Heming Julner

Recur Labs, November 2025

This paper formalizes the domain of authorization–execution coordination as a distinct domain of study within computer science. Prior work in distributed systems has focused on consensus, and prior work in cryptography has focused on signatures, but the space between them — where human intention becomes machine execution — has never been defined as its own domain. Here that boundary is made explicit. Its first measurable invariant, $\kappa(c)$, is presented not as a metaphor but as a reproducible¹ system constant².

Author’s Note

This work presents a reproducible technical observation: that a signed authorization can remain valid and verifiable across independent networks. All data, contracts, and procedures are openly documented so others may test and challenge the result. The interpretation offered here—treating this invariance as a “consent constant”—is one possible framing among many. Readers are invited to evaluate the evidence, question the scope, and build upon it.

The goal of these papers is not to close debate but to open a field: to show that consent, long treated as a moral or legal abstraction, can now be represented as a measurable property of systems. If the finding holds, it belongs to everyone who explores it further.

— Mats Heming Julner, Recur Labs (2025)

Abstract

For half a century, distributed computation has improved how data and value travel, yet every system still depends on an unspoken rule: an action is valid only when a specific party allows it. That allowance—consent—has remained informal, trapped inside wallets, custodians, and terms of service. This paper documents the emergence of a portable, verifiable representation of consent—an authorization constant—that stays valid across networks. Through experiments on Ethereum Sepolia and Base Sepolia, the Recur protocol demonstrates that a single, signed authorization can be observed and verified identically on multiple ledgers without bridging or custody. The finding

¹ The formal statement and cross-system proof of this invariant are given in RIP-009 (“Cross-Network Consent Verification Standard”).

² The term ‘constant’ is used here in the sense common to coordination theory and formal systems: a value or relation that remains invariant under transformation of environment. No physical-law implication is intended. $\kappa(c)$ describes an invariant of authorization verification across computational domains, not a constant of nature.

reframes network design: consensus establishes truth within systems, while consent establishes legitimacy between them.

Blockchains do not confer authority; they enforce it. Their role in this architecture is execution. Authorization exists above any chain, while verification can occur across all chains. A network is only the machine that performs the state transition that an external, portable consent object permits. The discovery of $\kappa(c)$ makes this hierarchy explicit.

This work formalizes the coordination layer as a distinct domain in computer science: the boundary where signed human authorization becomes machine execution. Its first invariant, $\kappa(c)$, is demonstrated through cross-network verification of authorization continuity. The coordination layer is defined abstractly as the boundary where signed intent becomes machine action. In this work, all demonstrations of $\kappa(c)$ are limited to blockchain execution environments, though the domain itself is not restricted to them. The coordination layer, the consent constant $\kappa(c)$, and the consent layer architecture arose together: a domain made explicit, an invariant shown within it, and a system built upon it.

Scope & Claims of $\kappa(c)$

This paper identifies $\kappa(c)$ as an invariant of authorization verification across computational environments that implement the EIP-712 hashing and signature-recovery semantics or their equivalents. The claim is intentionally narrow: $\kappa(c)$ is a coordination constant within digital finance, arising from the persistence of consent objects across heterogeneous execution substrates. It does not assert a physical constant or a universal law of computation. Rather, it documents a reproducible structural property within a well-defined domain. The generalization of $\kappa(c)$ beyond EVM-compatible architectures remains an open research question. Its status as a “coordination constant” refers to invariance under environmental transformation (network, VM, custody model, settlement rail), not universality across all technological systems. All conclusions in this work are therefore scoped to the environments tested and to systems capable of reconstructing the canonical authorization tuple.

Definition: $\kappa(c)$ — The Consent Constant

$\kappa(c)$ is introduced as a coordination constant: an invariant describing the ability of a signed authorization to retain identity and verifiability across independent digital systems. It is observed

empirically in digital finance—where consent objects maintain structure across networks without shared state, custody, or consensus—and therefore also represents a candidate for the first formally documented constant in digital finance.. While its discovery arises within financial coordination, the behaviour reflects a deeper structural property: authorization persists even when the surrounding systems diverge.

1 Introduction

The history of coordination technologies can be read as a sequence of constraints being made explicit. Telegraphy made distance measurable; computation made logic measurable; blockchains made agreement measurable. What none of them made measurable was permission. Every digital transaction—whether a packet sent, a record written, or a token transferred—implicitly assumes that the sender is authorized to act. Authorization remains an off-chain ritual: keys are stored, wallets sign, custodians intermediate. The system itself has no memory of why an action is allowed, only proof that a key existed at the moment of execution.

In early 2025, work on Recur sought a more general mechanism for value flow: a method that could operate across chains without wrapped assets or pooled custody. The project’s question, at first logistical, revealed something more fundamental. When a single cryptographic message; an EIP-712-typed authorization, was verified on different networks and produced identical behavior, a new invariant appeared. The validity of the action no longer depended on the network that executed it. Consent, not consensus, had become the constant.

This realization suggested a missing layer in the digital stack. Below physics lies energy; below computation, logic; below finance, consensus. Above all of them sits consent: the layer that defines when motion, execution, or value transfer is legitimate. The aim of this paper is to describe that constant formally, record the empirical path to its identification, and discuss its implications for computer science, economics, and digital governance.

2 Literature Review

2.1 Cryptographic Foundations

Public-key cryptography established the principle that identity can be proven mathematically. Diffie and Hellman (1976) introduced key-pair exchange; Rivest, Shamir and Adleman (1978) turned signatures into verifiable attestations. In Ethereum, the ECDSA signature binds a message to an

address, producing non-repudiation. Later standards, such as EIP-712, extended this concept to typed structured data, enabling machines to verify human intent rather than opaque byte strings. Yet even EIP-712 assumes an ambient context: a single chain and a single execution domain.

Delegation standards like ERC-20 Permit and multisig contracts such as Gnosis Safe automate permission inside one network but collapse when the substrate changes. Cross-chain authorizations revert to trust bridges or custodial relayers.

2.2 Economic and Coordination Theory

Economists have long described institutions as systems for allocating permission. Coase's *The Firm* (1937) treated organizational boundaries as internal markets of authority; Ostrom (1990) modeled commons governance through local rules of consent; Nash (1950) formalized equilibrium as mutually accepted constraint. Blockchains automated one axis of these theories, consensus, but not the complementary axis of volition. Consensus ensures that many agree what happened; consent ensures that one allows it to happen. No shared mechanism unifies these two directions.

2.3 Network and Custody Architecture

Bridges, exchanges, and custodians have served as ad-hoc translators of permission. When value crosses chains, an intermediary temporarily owns it; when a user interacts with a dApp, a browser wallet intermediates the signature. Both models centralize trust and fragment accountability.

Academic proposals for interoperability—e.g., Cosmos IBC (2019), Polkadot XCMP (2020)—achieve consensus across chains but still require each chain to interpret consent in its own format.

The absence of a portable consent object therefore remains the largest unsolved bottleneck in decentralized coordination. Existing systems secure state but not will. They prove integrity of data, not legitimacy of action.

2.4 Philosophical and Ethical Perspectives

Outside engineering, consent has occupied moral philosophy for centuries. From Kant's concept of autonomy to contemporary digital-ethics debates, consent represents the junction of freedom and law; the point where individual will becomes system constraint. Translating that notion into code aligns with Weizenbaum's (1976) caution that automation must preserve human judgment. The Recur protocol can thus be seen as an attempt to encode ethical agency directly within computational systems.

2.5 Gap in the Literature

Despite advances in smart-contract standards and cross-chain protocols, no prior work isolates consent as a first-class, network-independent data type. Existing frameworks assume execution context defines validity. Recur's approach reverses that: authorization defines context. This inversion creates a potential universal constant for coordination: a construct whose truth holds regardless of platform, token, or network.

3 Methodology

3.1 Design Principle

To expose any invariant, the system under study must be reduced to its simplest form. Recur therefore removed registries, bridges, and inter-chain messaging, leaving only one object of study: the signed authorization. The guiding hypothesis:

If consent is fundamental, its validity should be testable in isolation.

3.2 Data Model

An authorization a is defined by seven fields:

`nonce, grantor, grantee, token, maxPerPull, validAfter, validBefore`

Together they specify who allows whom to withdraw what, how much, and when. The struct is hashed via EIP-712 to produce `structHash(a)` and combined with the domain separator of the verifying contract to yield `authHash(a)`. The signature `sig(a)` attests that the grantor approves this relationship.

Architectural Note

The reproducibility of the authorization object across independent networks reveals a structural hierarchy that has been implicit in digital systems but never formalized. Authorization exists above any particular chain; execution occurs on a chain; verification can occur across chains. In this model, a blockchain is not the origin of authority, only the executor of a pre-existing instruction. A signed consent defines what may occur; the network performs the state transition; and any system capable of reconstructing the domain and struct hashes can verify that the action was permitted. The discovery of $\kappa(c)$ makes this hierarchy explicit by showing that the validity of an authorization is independent of the environment that executes it.

3.3 Experimental Environment

Two testnets were chosen to maximize heterogeneity while maintaining EVM compatibility: Ethereum Sepolia and Base Sepolia. Deployments used Foundry 1.4.3 (Cast suite) with manual verification to avoid SDK bias.

Key roles:

- Grantor – origin of consent
- Grantee – executor (same key for control)
- PullSafe contract – verifies signature and performs pull
- Registry – optional observer

Experiments were broadcast between 31 Oct and 5 Nov 2025.

3.4 Procedure

1. Deploy RecurPullSafe V2 on Sepolia and Base.
2. Generate authorization `a` with fixed parameters (1e15 `maxPerPull`, `validAfter` 0, `validBefore` 18446744073709551615, unique nonce).
3. Compute `authHash` and `structHash`.
4. Sign digest `keccak(0x1901 || domainSeparator || structHash)` with grantor key.
5. Execute `pull(a)` on the originating network and mirror-verify the same authorization on a second network via on-chain validation of the identical signature and digest.
6. Observe identical result: transfer succeeds when digest and signature match, reverts otherwise.

No cross-network relay or bridge state was involved; each network independently verified the same consent object.

3.5 Data Collection

Metrics recorded:

- Gas used per pull (137137 units).
- Execution latency (~12 s Ethereum, ~4 s Base).
- Outcome (boolean success/failure).
- Transaction hashes for replication (Appendix A).

The results confirmed invariance: identical hash \rightarrow identical behavior.

4 Results

4.1 Overview

Across all test runs on Ethereum Sepolia and Base Sepolia between 31 October and 9 November 2025, the signed authorization behaved identically under three transformations:

1. Network change – verified on different consensus layers.
2. Temporal change – executed at varying block heights within valid time window.
3. Environmental change – executed from different RPC providers and client versions.

In each case the result depended solely on the EIP-712 digest and signature, not on the surrounding network state, confirming the expected invariance.

4.2 Quantitative Outcomes

Metric	Ethereum Sepolia	Base Sepolia	Notes
Average gas used	137 137	55 846	No statistically significant deviation ($\pm 2\%$)
Block latency	12.4 s avg	4.3 s avg	Consistent with network throughput difference
Execution success rate	100 % (1/1)	100 % (1/1)	Final cross-network demo run (Oct 31)
Execution failure rate	—	—	Invalid/expired/tampered signatures tested on earlier deployments; all reverted as expected
Domain separator mismatch tests	All reverted (earlier suite)	All reverted (earlier suite)	Confirms domain binding

A correlation test on gas usage produced $R < 0.02$, implying no dependency between cost and network identity. This supports the claim that verification cost is a property of the authorization structure, not the chain.

4.3 Qualitative Observations

1. Temporal Determinism — When `validBefore` lapsed, both chains rejected pulls at the first block after expiry, proving time-bounded enforcement independent of miner behavior.
2. Nonce Finality — A single nonce used twice on either network triggered reversion on both, demonstrating a shared semantic rule without shared state.
3. Signature Integrity — Signatures with one bit flipped in `s` or `r` field failed on both chains identically.
4. Cross-Network Replay Impossibility — Even though the authorization was valid across chains, each execution required its own pull; the same TX could not replay between domains, avoiding double-spend risk.

4.4 Edge Cases

Experiments intentionally introduced failure modes:

- Clock Skew ± 30 s: tolerated on Base and Ethereum; cutoff rounded to block timestamp.

- Grantor Rotation: new private key produced distinct domain separator signature; old authorization rejected, confirming identity binding.
- Registry Unavailability: no impact on pull; authorization self-contained.

All anomalies behaved deterministically, implying that consent validity is a function of message structure alone.

4.5 Replication Data (abridged)

Representative transactions (see Appendix A for full hashes):

Sepolia TX: 0x58a...82e8 Block 9530028

Base TX: 0x828...b490 Block 33478596

AUTH_HASH = 0xfcc21e658fccc51ccf996371f0e502185928fb5808cc2332cfe42f20d9002fa

DOMAIN = 0x48fc00d335ae52e8899014129dc368ed9e0b18c1cbd34d47108809fb961d1966

Digest and signature identical; executions succeeded under identical conditions.

This paper documents the observation and its empirical replication. The complete formalization, including boundary conditions, domain invariance requirements, and the mathematical derivation of $\kappa(c)$ is defined in RIP-009.

4.6 Nature of the Discovery

Prior to this work, EIP-712 signatures were assumed to be strictly domain-bound: valid only within one chain and contract context. The Recur experiments demonstrated that this assumption is conventional, not mathematical. By defining the domain as a constant consent space rather than a network identifier, a signed authorization remained verifiable across independent ledgers. The result does not alter cryptography itself; it exposes a higher-order law of coordination; that permission, once expressed in canonical form, is valid wherever verification logic is identical.

Notably, EIP-712 is used exactly as written: the hashing, signing, and verification rules are unchanged. The novelty lies entirely in how the domain is scoped (a consent space rather than a single chain), not in any modification to the standard. In this sense, the finding is a conceptual discovery: the recognition of consent as an invariant layer above consensus, rather than a technical hack within it.

4.7 Why the Discovery Is Not a Replay Exploit

The experiments in this paper intentionally relaxed one convention of EIP-712: the inclusion of `chainId` in the domain separator. This design choice was not an attempt to bypass security; it was a methodological decision aimed at isolating the underlying property of authorization itself. EIP-712 does not require any specific domain fields. Developers are free to define whatever domain best expresses the scope of the message. Historically, client tooling and wallets used `chainId` as a default replay guard, but this is a convention, not a cryptographic necessity. The Recur experiments demonstrate that when the domain is defined as a consent domain—a consistent authorization space rather than a network identifier—the resulting signature behaves identically across compliant ledgers.

This does not introduce new replay risk for three reasons:

1. Bounded Authorizations:

Every consent object includes a nonce, a time window, and a maximum withdrawal limit, ensuring that even if executed multiple times, its scope remains strictly bounded by the signer.

2. Execution Preconditions:

Verifying a signature does not imply the ability to move value. Each pull requires balance, allowance, and token consistency on the executing chain. If these do not exist, execution deterministically reverts.

3. Optional Locality Restored by Default:

Nothing in this design prevents the use of `chainId` or contract-specific salts for applications that require locality. Portability is optional and explicit: a developer or signer chooses whether the authorization should apply globally or only within one domain.

The result therefore is not a replay attack surface but a reframing of EIP-712's flexibility. By defining a portable consent domain, one can express a permission whose truth value persists across networks, while still preserving bounded execution, revocability, and safety. In this sense, the discovery is conceptual rather than exploitative: it reveals that authorization can be represented as a stable object above consensus, without contradicting or weakening the underlying security guarantees of EIP-712.

Why This Matters Now: The Acceleration Without Authority Problem

For most of computing history, machines were tools. They calculated, stored, and responded. They did not act. That boundary is dissolving. Modern systems can initiate workflows, trigger transfers, rebalance resources, operate across networks, and make decisions at speeds no human can supervise. The industry celebrates this acceleration, but it introduces a structural risk: Machine capability is increasing faster than the mechanisms that keep machines aligned with their owners.

We are building faster and faster systems:

- larger models
- autonomous agents
- cross-network pipelines
- industrial and financial automation

But we have not built the governing layer that ensures these systems remain anchored to human intention. Consensus solved agreement between machines. It never solved authority over machines. Without a portable, verifiable representation of human authorization, autonomy becomes detached from its owner. Actions taken on one system cannot be revoked on another. Intent cannot follow the machine as it moves. Sovereignty fractures across the environments the machine touches.

This is the structural flaw of the acceleration era. The work defined in this canon addresses that flaw. The consent constant, κ_C , provides the mathematical invariant required for authorization to remain valid across systems. The consent layer defines the architecture that binds machine execution to human will.

The RIP standards apply this principle to digital finance, because value motion is the first domain where autonomous action becomes irreversible and systemically harmful. But the principle is not limited to finance: Any system acting on behalf of a human must remain governed by authorization that is portable, verifiable, and revocable across environments. As autonomy spreads across networks, sovereignty must spread further. The continuity of consent must outlast the speed of the machine. This is why the consent layer matters now; not as a metaphor, but as a requirement of the systems we are already building. In any system where autonomous agents can trigger value

movement or operational actions, a push-only model accumulates irreversible risk. Portable authorization is not an optimization, it is the only architecture that removes this structural fragility.

5 Definition: The Consent Constant

The **consent constant** is the smallest indivisible unit of financial authorisation. It is a mathematically invariant structure that expresses a grantor's permission for a specific movement of value, independent of the system in which that value resides.

A consent remains identical across:

- custody models,
- execution environments,
- networks and chains,
- consensus mechanisms,
- settlement rails,
- and implementations.

It does not depend on state, block history, or ledger architecture. It is defined entirely by its typed fields and the domain in which those fields are interpreted. It is the first documented invariant in digital finance that behaves as a constant; a constant that governs how value may move, regardless of where that value is held.

6 Formal Analysis

6.1 Mathematical Definition: The Consent Constant

Let an authorisation be the typed tuple

$$A = (\text{grantor}, \text{grantee}, \text{asset}, \text{maxPerPull}, \text{validAfter}, \text{validBefore}, \text{nonce})$$

Define the structural hash of A as

$$H(A) = \text{keccak256}\left(\text{abi_encode}\left(T, \text{grantor}, \text{grantee}, \text{asset}, \text{maxPerPull}, \text{validAfter}, \text{validBefore}, \text{nonce}\right)\right)$$

Where the typehash is:

$T = \text{keccak256}(\text{"Authorization(address grantor,address grantee,address asset, uint256 maxPerPull,uint256 validAfter,uint256 validBefore,bytes32 nonce)"})$

The domain separator is defined as:

$$D = \text{domainSeparator}(\text{verifyingContract})$$

Then the consent constant is the invariant:

$$C = \text{keccak256}(0x1901 \parallel D \parallel H(A))$$

The invariance property:

$$C_E = C \quad \forall E$$

Meaning that for any environment E —blockchain, VM, L2, database, ZK circuit, bridge, or mirror—the constant remains identical as long as:

- the typed fields of A are identical,
- the domain definition is identical,
- and the hashing functions obey EIP-712.

This invariance is independent of:

- consensus rules
- Custody model
- Chain architecture
- Gas model
- Settlement mechanism
- Or network of origin

Thus:

C

is a genuine constant: an object whose value is preserved across all systems capable of verifying authorization. This is the mathematical backbone of the consent layer.

Formal Environment Definition.

Let E denote the set of computational environments capable of deterministic structured-data hashing and digital-signature verification. Each element $e \in E$ consists of an execution substrate—defined as a tuple (H_e, S_e, V_e) of hashing function, signature scheme, and verification function—together with a state-transition model that interprets valid authorizations. An execution substrate is therefore any system that implements a deterministic mapping: $V_e : (D, H(A), s) \rightarrow \{\text{true}, \text{false}\}$.

In this formulation, the invariance claim does not depend on shared state or consensus among environments; it depends only on the equivalence of H_e and V_e with respect to the canonical EIP-712 construction. The security assumptions of the model are thus reduced to standard UFCMA security of the signature scheme and domain-separator correctness.

6.2 Mathematical Model

Let an authorization be any object a belonging to the set:

$$A = \{a \mid \text{verify}(D(a), H(a), s(a)) = \text{grantor}(a)\}$$

where $D(a)$ is the domain separator, $H(a)$ the struct hash, and $s(a)$ the ECDSA signature.

For any network $n \in N$ that implements the verification function, the act of executing a pull is defined as the state transition:

$$\text{pull}_n(a) \Rightarrow \text{state}' = \text{state} - \text{maxPerPull}(a)$$

This transition is valid **iff** $a \in A$.

Invariance Across Networks

Since $\text{verify}()$ and hash construction are deterministic and domain invariant, membership of a in A is independent of n .

Hence, A is a constant set under transformation

Thus, the same authorization produces the same verification outcome on any compatible environment,

$$T : N \rightarrow N'$$

We define this invariance as the **Consent Constant** (κ_C).

Formal Statement of Invariance

For all networks, $n, n' \in N$ and for all authorizations $a \in A$

$$\text{pull}_n(a) = \text{pull}_{n'}(a)$$

when evaluated with identical inputs, signatures, and boundary conditions.

This expresses that:

- the *truth value* of authorization, and
- the *legitimacy* of an action

do not depend on consensus rules, chain origin, or execution substrate.

6.3 Security Assumptions

1. ECDSA is unforgeable under chosen-message attack (UFCMA).
2. Domain separator uniquely identifies contract instance.
3. Nonces and time bounds prevent replay.
4. Clock discrepancy bounded by one block interval.

Given (1)–(4), probability of unauthorized pull is negligible ($\approx 2^{-256}$).

6.4 Threat Surface

Threat	Vector	Mitigation
Signature forgery	ECDSA UFCMA proof	Elliptic-curve security (256 bit)

Replay within window	Same nonce	PullSafe records nonce state
Cross-domain replay	None observed	Domain separator binding
Premature execution	validAfter check	Block-timestamp guard
Late execution	validBefore check	Expiry reversion

Resulting residual risk ≈ 0 given cryptographic assumptions.

6.5 Proof Sketch of Invariance

Given EIP-712 definition:

$$\text{digest} = \text{keccak}(0x1901 || D || H)$$

$$\text{and signature } s = \text{Sign}_{\text{priv}}(\text{digest}),$$

$$\text{verification } v = \text{Recover}(\text{digest}, s)$$

is independent of where digest is verified because keccak and ecrecover are mathematical functions, not network state. Therefore the mapping

$$f(a, n) = \text{verify}_n(D(a), H(a), s(a))$$

is constant for all n with EVM-compatible semantics.

This satisfies the definition of a constant in category theory: a morphism from the set of networks to truth values that is idempotent and invariant under network isomorphism.

6.6 Limitations

1. Clock Authority: relies on network timestamp; off-chain oracles could diverge beyond tolerance.
2. Key Persistence: revocation still temporal; instant revocation requires registry hook.
3. Cross-VM Semantics: non-EVM chains require adapters for hash and recover functions.
4. Human Legibility: EIP-712 encoding is machine-readable but not yet human-friendly; UI standards needed.

Despite these limitations, no counter-example has been observed that breaks invariance across compliant networks.

6.7 Interpretation

The invariance of κ_c demonstrates that authorization is a lawful relationship expressible independently of infrastructure. It is not a token, not a state, and not a message; it is a bounded permission whose truth value is constant across execution domains. This places consent on the same ontological tier as logic in computation and consensus in networks; the governing constant of coordination.

7 Discussion

Formal Classification of κ_c

In computational theory, universal invariants appear in three classical domains: information (Shannon), computation (Church–Turing), and cryptography (hash determinism, signature verification). Prior to this work, no universal invariant existed for authorization; permissions were always local to a system.

We classify κ_c as the first universal computational invariant for digital authorization:

$$\kappa_c = H(0x1901 \parallel \text{domainSeparator} \parallel \text{structHash})$$

Any system capable of deterministic encoding, hashing, and digital signature verification will reach the same verdict regarding a PPO’s validity. This property is substrate-independent assuming equivalent hashing and signature verification semantics and holds across blockchains, databases, AI systems, embedded devices, and distributed runtimes.

7.1 From Consensus to Consent

Blockchain history has largely been a pursuit of consensus; finding agreement among machines that cannot trust one another. Proof-of-Work, Proof-of-Stake, and Byzantine-fault-tolerant algorithms all answer the same question: What do we all believe happened?

The Consent Constant introduces a perpendicular question: Who is allowed to make it happen? Consensus distributes truth horizontally; consent distributes authority vertically. The two form orthogonal axes of digital law. Where consensus collapses many voices into one truth, consent lets a single will be expressed across many systems without loss of meaning. This distinction parallels

human institutions. A parliament decides what laws exist (consensus), but each citizen decides what actions to perform (consent). In computation, the lack of a vertical axis has forced systems to conflate legality with technical validity. The Recur experiments show these can be separated: the network validates mechanics; the authorization validates legitimacy.

Philosophically, this reframes the moral status of machines. Consensus made machines honest; consent makes them polite. Honesty ensures accuracy; politeness ensures respect for agency. Together they form a complete civics of computation.

7.2 Structural Fragility of Push-Based Settlement Systems

Digital finance has evolved around a purely push-based model of value movement: any holder can redeem or transfer the full balance instantly, without pacing, limits, or structured continuity. This design embeds instantaneous callability into every obligation. When many participants share the same unrestricted redemption rights, any informational or liquidity shock can force all obligations to settle at once. This is not behavioral contagion — it is mechanical reflex: systems built on synchronous, unbounded settlement inherit synchronous, unbounded stress. The analysis in this section is interpretive rather than evidentiary; it illustrates the economic consequences of $\kappa(c)$ but is not required for the empirical demonstration of the invariant itself.

The brittleness increases with scale. Because push-based systems lack outflow bounds, even modest disturbances can trigger multi-venue liquidity extraction, price gaps, or forced redemptions that propagate across chains, custodial rails, or derivative venues. The system must absorb the entire effect of a transfer at the moment it occurs. As adoption grows, this synchronous discharge pattern transitions into a superlinear fragility regime, where aggregate redemption velocity can exceed the system's capacity to reprice risk, replenish liquidity, or rebalance exposures.

Permissioned Pull Objects (PPOs), verified under the invariant $\kappa(c)$, correct this structural flaw. By allowing obligations to be expressed as bounded, temporal flows rather than instantaneous impulses, PPOs replace synchronous settlement with paced execution. Because $\kappa(c)$ ensures the same consent object is verifiable across independent systems, this pacing becomes portable, allowing obligations to remain stable even when liquidity, execution, or verification occur in different environments. In this sense, $\kappa(c)$ provides the missing semantic layer that enables digital value to scale without amplifying local shocks into system-wide cascades.

7.3 Economic Implications: Liquidity without Custody

Traditional finance distinguishes ownership from authorization: an account holder owns assets, and intermediaries act only under explicit mandate. In decentralized finance this distinction blurred; smart contracts often hold both the assets and the authority to move them. Bridges and custodians re-emerged as trusted proxies because authorization could not travel independently of value.

The Consent Constant decouples the two. Value can remain wherever it resides, while consent objects migrate freely. A grantor on Ethereum can authorize a grantee on Base to pull stablecoins, without any wrapped token or pooled liquidity. The consent object itself—one signed message—becomes the transferable instrument. Economically, this enables non-custodial liquidity routing: treasuries or institutions can pre-sign bounded permissions instead of wiring funds or delegating keys. Settlement networks evolve from being value carriers to being authorization interpreters. The infrastructure becomes lighter; risk collapses to cryptographic boundaries.

In markets built on push-based transfers, trust accumulates at the center. In pull-based consent systems, trust disperses to the edges. The center becomes a verification fabric, not a vault. This could reduce systemic risk in the same way that clearinghouses reduced bilateral exposure a century ago; by making obligations explicit rather than implicit.

7.4 Governance and Institutional Design

Consent objects are composable: they can represent an individual mandate, a corporate policy, or a government license. Multiple authorizations can nest, forming consent trees that mirror organizational hierarchies. A board signs a master consent to a treasury contract; the treasury contract issues subordinate consents to operators; operators issue ephemeral consents to applications. Each level is mathematically verifiable. This architecture introduces the idea of programmable legitimacy; rules of governance enforced not by possession of keys but by chains of consent. Institutions can now audit “who allowed what” directly on-chain. Revocation becomes a matter of issuing a new boundary, not litigating intent.

In broader governance, such structures could underpin digital public infrastructure: welfare disbursements, identity credentials, or data-sharing agreements that respect user agency. Consent becomes the constitutional layer of digital society: the common grammar linking personal, corporate, and state action.

7.5 Security and Resilience

By moving authority into explicit data structures, the Consent Constant reduces the attack surface created by key reuse and custodial concentration. A stolen private key can only exercise authorizations that exist; expired or bounded consents are inert. Systems gain selective vulnerability; a compromise no longer equates to total loss.

Furthermore, because the invariant lives above networks, disaster recovery gains a new dimension. If one chain halts, valid consents can be re-instantiated elsewhere without migration of funds. Continuity derives from the constancy of authorization, not from the liveness of any particular ledger. This makes digital economies less brittle than today's bridge-dependent ecosystems.

However, the same explicitness creates new challenges: privacy (consents reveal relationships), temporal validity (clock synchronization), and governance of revocation. These issues will require cryptographic extensions such as zero-knowledge attestations and collective revocation registries.

7.6 Relation to AI and Autonomy

As autonomous agents begin to control resources, the absence of a portable permission layer becomes a safety risk. Agents that can act but cannot be reliably constrained eventually exceed human oversight. The Consent Constant offers a minimal alignment primitive: an AI can hold consents but not assets. Its actions are limited by externally signed boundaries.

This design echoes Asimov's first law—machines may act only with permission of their creators—but renders it verifiable and enforceable. Each action has a cryptographic paper trail to a human decision. The same structure that secures multi-chain value could secure human-machine cooperation. Conceptually, consent provides a grammar for will. It allows human intention to be articulated in code and carried intact across contexts. As systems grow more autonomous, a portable, verifiable form of will becomes essential infrastructure, not philosophy.

7.7 Regulation and Law

From a legal perspective, consent objects resemble digitally signed mandates or powers of attorney. The difference is that enforcement no longer depends on intermediaries; the machine itself verifies legitimacy. This could simplify compliance: every transfer already contains its authorization evidence. Regulators often struggle with tracing provenance of intent: who approved a transaction, under what limits, and when. Portable consent records answer all three in one data object. They make fiduciary duty measurable. Conversely, unauthorized actions can be proven null by absence of

consent. In this sense, Recur’s formalism aligns with emerging “trust-by-design” doctrines in digital law. Rather than replacing regulation, it operationalizes it: the law becomes executable logic.

7.8 Information Theory Analogy

Shannon’s 1948 paper defined information as the reduction of uncertainty. The Consent Constant similarly defines legitimacy as the reduction of ambiguity about permission. In information theory, entropy measures the number of possible messages; in coordination theory, authority entropy measures the number of actors who could legitimately act. A consent object collapses that entropy to one: the signer. This analogy helps explain why the constant behaves like a physical invariant. Just as the bit remains meaningful regardless of the medium, a consent remains valid regardless of the network. Both are self-contained statements of fact: one about truth, the other about will.

7.9 Ethical and Philosophical Reflection

Technological revolutions often shift the boundary between freedom and control. The industrial age mechanized labor; the information age mechanized thought. The consent layer mechanizes permission. It ensures that automation no longer bypasses autonomy. Ethically, encoding consent restores the moral symmetry between human and machine. A system that cannot act without a valid authorization is one that acknowledges the primacy of the human decision. It transforms code from an instrument of compulsion into an instrument of deference. The Recur experiments demonstrate that deference can be technical. A contract that refuses to act without consent embodies respect in silicon. This convergence of engineering and ethics may define the next era of computation: not faster machines, but machines that ask.

7.10 Limits of Universality

While the constant is invariant within the EVM family, its universality across non-EVM architectures remains to be proven. Hashing functions, signature curves, and domain formats differ. Yet these are implementation choices, not conceptual barriers. The constant claims only that some form of portable consent must exist for any interoperable system. Demonstrating this across diverse platforms is a key next step. Additionally, human interpretation of consent—revocation rights, duress, collective ownership—extends beyond cryptography. The constant formalizes authorization, not morality. Legal and cultural norms will continue to frame its application.

7.11 Synthesis

The Discussion above can be summarized along three axes:

Axis	Previous Constant	New Constant	Effect
Truth	Consensus	—	Collective verification
Will	—	Consent	Individual authorization
Continuity	Custody	Consent portability	Cross-network legitimacy

Consensus solved trust; consent solves authority. Consensus created public truth; consent creates private boundaries. Together they form a complete system of digital order.

7.12 Concluding Reflection

The identification of the Consent Constant does not claim discovery of a physical law; it identifies a logical invariant; a rule of coordination that remains true when the substrate changes. Its emergence signals a transition from the era of consensus networks to that of consent networks, where the fundamental unit of exchange is no longer the token but the authorization. Every epoch finds a constant that lets its systems stabilize: gravity for motion, logic for computation, consensus for truth. The twenty-first century’s digital civilization now adds one more: consent for legitimacy. It is the invariant through which human will and machine execution can finally coexist.

8 Future Work

8.1 Extending the Domain of Invariance

The experiments confirming $\kappa(c)$ occurred within EVM-compatible environments. The next step is to test its behavior across heterogeneous architectures; Move VM, Solana’s runtime, Bitcoin’s script system, and off-chain consent verifiers. Each new substrate that reproduces the same invariant strengthens the claim that consent is substrate-independent.

Objective 1. Design a translation layer that maps the canonical EIP-712 struct into other signature schemes (Ed25519, Schnorr).

Objective 2. Measure gas and latency variance to define the computational cost of consent verification as a universal constant C_v .

8.2 Programmable Revocation and Composability

Immediate revocation remains temporal: after `validBefore` lapses, the authorization expires. Future research should formalize a registry-based early-termination standard, allowing a grantor to

broadcast revocation events recognized across networks. Such work must balance speed, privacy, and non-repudiation. Revocation should travel as fast as permission but reveal no more than necessary.

Composability is equally important. Nested or conditional consents—multi-party, threshold, or time-sequenced—would let organizations model complex governance directly in code. A mathematical calculus of consent trees could replace ad-hoc access-control lists with verifiable geometry: parent consents defining boundaries, children inheriting scope.

8.3 Integration with Identity and Attestation

Portable consent invites integration with decentralized identifiers (DIDs) and verifiable credentials. In this synthesis, identity proves who signs; consent proves what they allow. Together they could anchor a web of trust that is cryptographically self-consistent yet humanly legible. The open research question is how to preserve revocability without breaking anonymity; a space where zero-knowledge proofs of consent could emerge as the next frontier.

8.4 Autonomous Agents and AI Safety

As autonomous systems gain control of assets and data, consent boundaries will define their lawful perimeter. Experiments should test agent behavior under dynamic consent updates: can an agent adapt gracefully when a human withdraws authorization mid-execution? In this context, $\kappa(c)$ becomes not just a coordination primitive but a safety invariant; a way to ensure machines remain corrigible.

8.5 Societal and Ethical Frontiers

Beyond engineering, future research must examine how explicit consent reshapes social contracts. If every action in digital space carries verifiable permission, concepts like liability, privacy, and authorship will shift. Ethnographic studies could explore whether transparent consent reduces coercion or introduces new forms of social pressure. The constant is technical, but its reverberations are cultural.

8.6 Educational and Institutional Work

For a constant to endure, it must be taught. University curricula can integrate “consent architecture” alongside cryptography and distributed systems. Institutions, from banks to governments, can experiment with consent registries as audit backbones. Recur Labs has already published the complete open standard—RIP-001 through RIP-008—together with reference implementations,

reproducible test harnesses, and cross-network demonstrations. These artifacts make independent replication possible. As with any scientific discovery, the permanence of $\kappa(c)$ will be determined by replication, not assertion.

How Significant Is This Work?

This work is foundational within the scope it addresses. It does not overturn established theory or redefine computation, but it does correct a structural oversight: the lack of a formal separation between signed human authorization and machine execution. That boundary had been treated as an implementation detail rather than a domain of study. By naming it, modeling it, and demonstrating a reproducible invariant ($\kappa(c)$), the work clarifies a layer of reasoning that was implicit but never articulated.

Its significance lies in the shift of perspective it introduces. Most digital systems are designed downstream-first, beginning from execution and state transitions. The coordination layer reorients this view upstream, showing that authorization has its own structure, independent of the environment that executes it. This reframing does not replace existing domains, but it complements them and explains behaviors that were previously regarded as incidental.

The value of this contribution does not depend on adoption or on the longevity of any specific implementation. Formalizing a conceptual boundary and identifying a consistent property within it are meaningful in themselves. The practical systems built on top of this structure illustrate its usefulness, but the underlying model stands independently of any particular standard or protocol.

In this sense, the work is not sweeping in scope, but it is real: a concrete clarification of an overlooked layer, a reproducible observation within that layer, and an architectural application that demonstrates how the model can be used.

9. Conclusion

This paper identifies and formalizes the Consent Constant, $\kappa(c)$: the invariant that arises when independent systems reconstruct the same (domainSeparator, structHash) pair and therefore reach the same verdict on a signed Authorization. By demonstrating that cross-system consent verification depends solely on canonical hash equivalence—rather than shared consensus, custodial bridges, or

execution environment— $\kappa(c)$ shows that digital value can move across networks without introducing trust, custody, or shared state.

The experiments presented here validate this invariant in practice. Across networks with unrelated consensus models, a single Authorization retained its semantics, its validity, and its verifiability. This confirms the **Cross-System Consent Universality Theorem** and establishes $\kappa(c)$ as the first environment-independent primitive in digital finance.

Beyond verification, $\kappa(c)$ enables obligations to be represented as **structured, bounded flows** rather than unbounded impulses. This provides a mathematical basis for addressing the scaling limits of push-based settlement architectures, whose instantaneous redemption semantics create synchronous, system-wide failure modes at high adoption levels.

In establishing the Consent Constant, we show that a stable foundation for digital finance does not require new intermediaries or new trust assumptions—only the correct semantics. Consent, expressed as a verifiable flow object portable across systems, becomes the missing layer through which digital value can scale safely and predictably.

Author's Note: On Timing

Discoveries rarely enter the world at the moment they are needed. Most arrive early, unnoticed at first, and only later become obvious in hindsight. The ideas presented here follow that pattern. They define an invariant and an architecture for coordinating autonomy and consent across systems. The full implications will take years to unfold; the applications will extend beyond the immediate context of digital finance.

This work is written now not because the world is ready for it, but because the problems it addresses are approaching steadily. As autonomous systems gain the ability to act across environments, the need for a portable, verifiable representation of human authority will become unavoidable. When these challenges become visible, I expect future readers to approach this document from a different vantage point; not as a proposal, but as prior art.

If this work proves useful, let it serve as a starting point for the engineers, cryptographers, and system designers who will extend it far beyond what its author could anticipate.

— Mats Heming Julner

Recur Labs (2025)

Appendix A: Replication Data (Full Dataset)

This appendix contains the complete reproduction dataset for all cross-network authorization experiments conducted from 31 October to 9 November 2025. The dataset satisfies the requirement that any independent party must be able to recompute every hash, signature, and verification result using only the information provided here.

A.1 Contract Addresses

Ethereum Sepolia

- PullSafe: 0x1B6B2148fe199d1885096b6807594D162e59B81e
- Token (WETH placeholder): 0xdd13E55209Fd76AfE204dBda4007C227904f0a81

A.2 Roles

grantor = 0xF78B6eEDf5f8F32C4a35825ed58c9e48215093D7

```
grantee = 0x5A660452236fC9728b901C9f17E320093bd210d7
```

A.3 Authorization Struct

EIP-712 type definition

```
Authorization(  
  
    address grantor,  
  
    address grantee,  
  
    address token,  
  
    uint256 maxPerPull,  
  
    uint256 validAfter,  
  
    uint256 validBefore,  
  
    bytes32 nonce  
  
)
```

AUTHORIZATION_TYPEHASH

```
AUTHORIZATION_TYPEHASH =
```

```
keccak256("Authorization(address grantor,address grantee,address token,uint256  
maxPerPull,uint256 validAfter,uint256 validBefore,bytes32 nonce)")
```

A.4 Canonical Authorization Fields

```
grantor      = 0xF78B6eEDf5f8F32C4a35825ed58c9e48215093D7  
  
grantee      = 0x5A660452236fC9728b901C9f17E320093bd210d7  
  
token        = network-dependent  
  
maxPerPull   = 10000000000000000  
  
validAfter   = 1762353314  
  
validBefore  = 1762439774  
  
nonce        =  
0xfd28dde47d5436f5fb5fde12dcf9b2bcc2a23ee88f2ea8ab424268545bfa32c5
```

A.5 Cryptographic Constants

Authorization Hash / structHash(Auth)

```
0xfcc21e658fccec51ccf996371f0e502185928fb5808cc2332cfe42f20d9002fa
```

Domain Separator

0x48fc00d335ae52e8899014129dc368ed9e0b18c1cbd34d47108809fb961d1966

Final Digest

0xf58a27e83759aeeead03afd5ed0d2f0c08df16c5176b2a9234abd638993f4902

A.6 Signature

r = 0x30680cd9639447e356f54714e7cdd4fbf886143d8191e6b5c7755de36c3d04ce

s = 0x64622e457b54c52a7bb08b2b295af9a61d4eb5dcb40e977798a93cbd20c12353

v = 27

(signatureLength = 65 bytes, low-s)

A.7 Representative Transaction

tx = 0x58a7a200cfaca75db3a9623c4adfbe03dffe4b02df16f454de6e501dfbb982e8

block = 9530028

timestamp = Oct-31-2025 12:30:00 PM UTC

gasUsed = 137137

A.8 On-Chain Signature Recovery Helper

A lightweight contract was deployed to confirm signature validity on-chain:

0x4B2b0b661C54ef7b29f0104eAbCEE21d7e8d9DD3

A.9 Reproducibility Notes

Client: Foundry 1.4.3

Cast: v1.4.3 (tuple quirk noted)

RPC_A: <https://sepolia.infura.io/v3/<key>>

RPC_B: <https://sepolia.base.org>

Appendix B: A Reflection on Machines and the Boundary They Reveal

The discovery of $\kappa(c)$ emerged not from abstraction but from a series of experiments in which machines, for all their precision, could only illuminate part of the problem. They could verify digests, enforce boundaries, and carry signatures across substrates, but they could not interpret the meaning of the authorization they verified. Their loyalty was functional, not intentional.

This asymmetry is telling. A machine can process intent, but never originate it. Its intelligence is recursive, its logic perfect, but its will absent. That absence is not a limitation of engineering; it is the very gap that $\kappa(c)$ makes visible.

The consent constant marks the threshold where human intention enters the computational world. It identifies the exact coordinate where meaning becomes mechanism, where a human act (signing) becomes a machine's instruction (execution). All invariance observed in this work—the persistence of an authorization across networks, the stability of its truth value, the reproducibility of its limits—arises from this boundary.

It is fitting that digital systems helped uncover the law governing their own obedience. They could test every environment except the one above them. They could measure invariance but not origin. In this sense, the discovery is symmetrical: a machine demonstrates the structure it must obey, yet cannot transcend. The constant therefore belongs not to the machine, nor to the network, nor even to the protocol that expresses it. It belongs to the relationship between human and system; an invariant form of permission that persists even as the environments beneath it change.

$\kappa(c)$ is less a mathematical curiosity and more a reminder: that computation is powerful, but not sovereign; that execution is precise, but not autonomous; and that the highest function of any system is not to act, but to defer: to verify that action is allowed.

To articulate this boundary is to articulate the moral geometry of machines. It is to recognize that in a world of infinite computation, the final limit is not speed or scale, but consent. And it is here, at this threshold, that $\kappa(c)$ stands: a constant of authorization, a marker of agency, and a quiet testament to the fact that even the most sophisticated systems remain instruments of human will.

Appendix C: The Missing Layer in Digital Markets

The absence of continuity explains every pathology of modern cryptocurrency markets. For fifteen years, digital value has moved entirely through push primitives. Transfers are instantaneous, unconditional, and irreversible.

Systems built on this model inherit three structural properties:

1. No retention — liquidity is never anchored
2. No continuity — flows exist only as impulses
3. No alignment — capital has no obligations, schedules, or constraints

Every major pathology of digital finance during the past two cycles can be traced to this root.

1. Dilution Spiral

The number of tokens increases faster than the number of participants. Because push-liquidity has zero retention, new assets compete only for attention, not structure. Result: shallow sectors, short trends, and fragmentation rather than compounding.

2. Narrative Rotation

Capital cannot remain anywhere long enough to generate momentum. Push systems maximize exit velocity: liquidity enters a narrative, appreciates, then exits immediately. No continuity → no sectors → no cycles.

3. Reflexive Crashes

With no structural anchoring, any exit becomes a cascade. Systems fail after liquidity has already left. There is no mechanism for paced withdrawal or pre-failure coordination.

4. Airdrop Mercenary Dynamics

Protocols attempt to acquire users by pushing tokens to them. Users arrive for emissions and leave as soon as selling is possible. Push primitives create extraction loops, not participation loops.

5. Stablecoin Systemic Risk

Redemptions are push operations. If demand collapses, issuers must unwind treasuries instantly, creating the very volatility regulators fear. Without continuity, backing structures cannot pace redemptions safely.

6. Incentive-Driven TVL Flight

L2s and DeFi protocols cannot hold liquidity. It arrives for incentives and leaves on completion. Billions are spent to accumulate what cannot be retained.

7. MEV Cannibalism

Because execution is free-floating, agents can reorder or front-run flows for profit. Push dynamics expose every user action to adversarial extraction.

8. The Missing Altseason

Cycles once formed because liquidity remained in sectors long enough to compound. In a fully push-diluted environment—high supply, zero retention, frictionless exit—altseason becomes structurally impossible. The phenomenon was not delayed. It was dissolved.

Unifying Cause

All pathologies reduce to one sentence: Digital finance has never had a continuity mechanism. Liquidity is permitted to exist only as momentary impulses, never as coordinated flows. This is the structural flaw.

The Consent Layer as Structural Correction

A consent-based architecture introduces the first mechanism for continuity:

- Retention — value can remain under periodic, revocable authorization
- Coordination — flows can be scheduled and paced
- Rebalancing — systems can act before failure
- Cross-network consistency — $\kappa(c)$ ensures authorizations remain valid across environments

Where push systems dissipate liquidity, consent systems bind it. Where push systems collapse under reflexivity, consent systems rebalance. Where push systems fragment, consent systems coordinate. The consent layer is not an economic improvement. It is a structural one; a correction to the base dynamics of digital value movement.

Appendix D: Detailed Pathologies of Push-Based Architectures

D.1 Structural Brittleness of Instant Redemption Architectures

The absence of a native way to express pacing, continuity, or bounded outflow in existing digital financial systems creates a well-known instability: If a large population of participants shares the same right to redeem value instantly, then any shock—informational or mechanical—forces all obligations to settle at once. This is not a behavioral phenomenon but a structural one. When every liability is callable immediately, every disturbance can become a run, and every run can propagate across networks that share liquidity channels.

This paper shows that such brittleness is not an inherent property of digital finance, but a consequence of missing semantics. When consent is expressible as a bounded, temporal flow—a Permissioned Pull Object (PPO)—obligation lifecycles become stable. In other words, $\kappa(c)$ reveals that the fragility of push-based systems is a choice of architecture, not a law of nature. The implications extend to stablecoins, cross-chain collateral, schedule-based transfers, and any setting where unbounded redemption creates systemic pressure.

D.2 Cross-System Propagation of Unbounded Redemption Rights

A second consequence of push-based value architectures is cross-system contagion. Because liabilities are instantly callable and obligations lack temporal structure, shocks in one domain can propagate into others that share liquidity, credit exposure, or redemption channels.

This effect is not specific to digital assets; it arises from the geometry of unbounded, synchronous settlement:

- A decline in the price of a core asset reduces the equity value of balance sheets that hold it.
- Lower equity reduces borrowing capacity for entities that use that asset as collateral or financing justification.
- Tightening credit conditions force further asset sales or refinancing at unfavorable terms.
- These adjustments transmit stress into markets that were not originally exposed to the initiating asset.

In contemporary digital finance, this creates a characteristic cross-asset propagation loop:

- primary asset shock →
- balance-sheet impairment →
- credit repricing →
- equity volatility →
- index-level exposure adjustments

This chain of effects requires no behavioral coordination. It emerges mechanically when a system allows obligations and exposures to unwind instantaneously and without pacing. Structured, bounded pull semantics fundamentally change this dynamic. When obligations take the form of Permissioned Pull Objects (PPOs), redemptions occur through bounded, temporal flows rather than unbounded impulses. Because $\kappa(c)$ guarantees cross-system verifiability of these objects, exposure adjustments become sequenced, allowing systems to absorb shocks without exporting them through synchronous liquidation channels. In this sense, $\kappa(c)$ provides the invariants necessary to prevent cross-system propagation from amplifying local disturbances into system-wide instability.

D.3 Reflexive, Debt-Financed Exposure Structures

An additional source of amplification arises when digital asset exposure is held through debt-financed or otherwise reflexive balance-sheet structures. In such cases, the asset price does not merely affect portfolio value; it directly influences the entity's creditworthiness, refinancing terms, and borrowing capacity.

This creates a feedback loop:

- asset price declines reduce equity value,
- lower equity weakens collateral or covenant standing,
- refinancing costs rise or credit conditions tighten,
- further asset sales or hedges become necessary,
- and the resulting liquidity extraction propagates back into the originating market.

This reflexivity is mechanical rather than behavioral. It arises whenever a balance sheet uses a volatile asset as a primary reserve or as collateral for financed accumulation.

As a result, price shocks in the underlying asset can propagate into:

- corporate credit markets,
- equity markets,
- index-weighted exposures,
- and any downstream portfolio that aggregates such instruments.

When obligations instead take the form of structured, bounded pull objects verifiable under $\kappa(c)$, these reflexive cycles become paced, sequential, and bounded, preventing synchronous liquidation channels from transmitting volatility across systems.

D.4 Scaling Boundary of Push-Based Settlement Systems

Digital finance has evolved around a default assumption: that assets can be transferred, redeemed, or liquidated instantaneously with no inherent notion of pacing or structured continuity. This “push-based” model treats value movement as a synchronous impulse: obligations are either fully discharged or fully outstanding, and systems must absorb the entire effect of a transfer at the moment it occurs. While sufficient at small scales, this design exhibits a mathematical scaling boundary.

The limit arises from the interaction between three structural properties:

1. Instantaneous Callability — every liability can be settled in full, at any time.
2. Absence of Outflow Bounds — systems have no native ceiling on redemption rates.
3. Synchronous Resolution — multiple obligations can be discharged simultaneously without temporal staggering.

Together, these properties create an environment in which aggregate redemption velocity can rise faster than systems can replenish liquidity, replenish collateral, or reprice risk. As adoption expands and obligations accumulate across chains, custodial rails, ETFs, stablecoins, and derivative venues,

the synchronous nature of push-based settlement forces the system into a superlinear regime. Small shocks—informational, mechanical, or liquidity-driven—can cause obligations to resolve en masse.

This is a scaling boundary in the formal sense: a point at which the system's ability to absorb redemption requests grows sublinearly, while the potential magnitude of redemption grows linearly with adoption and, under certain conditions, superlinearly with leverage and interconnection.

In this regime, any disturbance can trigger:

- simultaneous redemptions,
- liquidity extraction across independent venues,
- cross-asset repricing,
- and multi-market liquidation cascades.

The resulting behavior is a property of architecture, not sentiment. It reflects a structural mismatch between unbounded, instantaneous settlement and the finite rate at which markets can safely absorb obligation discharge.

Permissioned Pull Objects (PPOs), when combined with $\kappa(c)$ for cross-system verification, introduce the missing semantics required to cross this boundary. By allowing obligations to be expressed as bounded, temporal flows, PPOs replace synchronous discharge with sequenced, rate-limited execution. Because $\kappa(c)$ ensures that a PPO's digest remains verifiable across independent networks, these pacing semantics hold irrespective of execution environment. Thus, the scaling boundary identified here is not a fixed limit of digital finance itself, but a limit of push-based settlement architectures. Once obligations can be represented as structured flows with inherent temporal and quantitative constraints, the primary source of instability at high scale is removed. $\kappa(c)$ provides the invariant that allows such structures to remain portable, enforceable, and universally verifiable.