Ecuación 0 en Sistemas de Pago: Reversión Superior al Consenso con Coste Marginal Nulo

Autor: Alejandro Hernández (Waynance) · Fecha: 29 de septiembre de 2025

Resumen

Presentamos la Ecuación 0 (E0) como un principio de arquitectura criptográfica que permite reversiones operativas (p. ej., devoluciones) por encima del consenso de una cadena base sin reescribir historia ni vulnerar la seguridad. “Por encima” significa que la decisión práctica de validez para el usuario y el comercio queda determinada por evidencias sucintas (pruebas zk/firma umbral) y compromisos económicos (colateral/slashing), de modo que la verificación honesta tiene coste marginal ≈ 0, mientras que cualquier intento de fraude sigue anclado al coste del 51% (o a romper supuestos criptográficos). La reversión es, pues, aplicativa y verificable, no un reorg.

1. Motivación

En pagos reales, la “reversión” deseada es devolver dinero o corregir un asiento tras la finalidad de L1. Reorganizar bloques está fuera de alcance y es indeseable. E0 busca: (1) irreversibilidad en L1 (seguridad económica) y (2) reversibilidad aplicativa (refund) con prueba y colateral, de verificación instantánea y barata. E0 entrega (2) sin tocar (1).

2. Modelo y Notación

L1: cadena con umbral económico μ (p. ej., 51%) y parámetro criptográfico λ.

Estado de pagos S: libro mayor aplicativo del comercio (off-chain o L2) con anclajes periódicos en L1.

Transacción original τ con hash $h = H(\tau)$, finalizada en L1.

Operación de reversión R: devolución/corrección referenciada a h.

Prueba sucinta π: zk-SNARK/STARK que certifica que R cumple la política P (p. ej., ventana temporal, artículo devuelto, límites KYC/AML, firma del merchant, idempotencia).

Firma umbral agregada σ: BLS t-de-n de un comité de riesgo/compliance.

Colateral c: garantía bloqueada (on-chain o custodial) sometida a slashing si se prueba un R inválido.

3. Ecuación 0 (Definición Operativa)

Llamamos Ecuación 0 a la propiedad límite del verificador honesto:

$\lim_{N\to\infty} C_{verify}(N)/N = 0$, con $C_{verify}(N)$ en $O(1)$ u $O(\log N)$, y con sonoridad (soundness) de pruebas/firma:

$\Pr[\text{aceptar R inválido}] \le \varepsilon(\lambda)$ (negligible).

Lectura: el coste marginal de verificar una reversión tiende a “cero operativo”, mientras que forzar una reversión inválida exige (i) capturar μ en L1, o (ii) romper los supuestos criptográficos (probabilidad $\varepsilon(\lambda)$).

4. “Reversión por encima del consenso” (metáfora formal)

R es “superior al consenso” porque su efecto práctico para las partes supera el mero hecho histórico en L1 sin reescribirlo: L1 conserva τ, y el sistema publica un estado corregido $S' = T(S, R)$ con evidencia sucinta π y/o σ, anclado con raíz Merkle o commitment en L1. Para usuarios y auditores, lo que cuenta es S' (saldo, abono, inventario), cuya verificación cuesta ≈ 0.

5. Protocolo R0 (Reversa Aplicativa con Prueba Sucinta)

Entradas: h, importe a, política P, colateral c. Salida: recibo de reversión $\Re=(h,a,\pi,\sigma,\rho)$, donde ρ es la prueba de inclusión en el acumulador/Merkle del estado S'.

Fases:

1) Commit de política y colateral: publicar Com(P) y bloquear c en L1.

2) Construcción de R: el merchant genera R=(h,a,meta) y la prueba zk π de que R ■ P; el comité firma R $\Rightarrow \sigma$ (BLS).

3) Actualización de estado y anclaje: aplicar $S' = T(S,R)$; publicar raíz Merkle root(S') en L1; emitir ρ.

4) Verificación cliente/regulador: $\text{Verify}_{zk}(vk,\pi)=1$, $\text{Verify}_{BLS}(\sigma)=1$ y ρ contra root(S').

Complejidad del verificador: O(1) (zk + BLS) y O(log N) (Merkle).

6. Teoremas de Seguridad

Teorema 1 (No-degradación): Si L1 es segura ante adversarios con poder $< \mu$ y π/σ tienen sonoridad $\varepsilon(\lambda)$ negligible, entonces $\Pr[\text{aceptar R inválido}] \leq \Pr[\text{L1 finaliza inválido}] + \varepsilon(\lambda)$.

Teorema 2 (Ventaja asimétrica E0): Definiendo ■ = C_forge / C_verify, con C_verify=O(1) y C_forge anclado a μ o a romper λ, se tiene ■ $\to \infty$ al crecer N.

7. Metáfora 0^0 como caso base universal

En combinatoria, el convenio $0^0=1$ evita coste extra en el caso base. E0 adopta esa metáfora: cada verificación honesta de R es “caso base” (comprobar una prueba/firma sucinta) con coste marginal ≈ 0, a diferencia del atacante.

8. Implementación de referencia (Waynance)

Política P: ventana de devolución, matching de SKU, KYC/AML para ≥ 999 €, límites diarios, idempotencia.

Prueba zk π: circuitos que atestan R ■ P sin filtrar datos sensibles.

BLS σ: comité rotatorio (VRF) firma cada R.

Estado S: libro mayor con raíz Merkle anclada cada t minutos en L1.

Colateral c: pool con slashing automático si se prueba fraude.

Recibo $\Re$: (h,a,π,σ,ρ) para el cliente.

Coste "$\approx$ nulo" del verificador: π (ms, KB), σ (1 verificación), ρ (Merkle corto).

9. Métricas objetivo (SLOs)

Latencia de verificación cliente: < 200 ms en móvil.

Tamaño de $\Re$: < 32 KB.

Anclaje periódico: 1–5 min.

$\varepsilon(\lambda)$: < 2^{-128}.

Tasa de disputas resueltas: 100% vía π + slashing.

10. Límites y alcance

E0 no reescribe bloques ni baja μ. La "reversión por encima del consenso" es semántica/aplicativa y auditada: el dinero vuelve al cliente y el estado contable se corrige sin tocar la historia. La seguridad de L1 permanece intacta; E0 abarata la verificación y eleva el coste del fraude.

Conclusión

La Ecuación 0 convierte la verificación de reversiones en un caso base de coste marginal $\approx$ 0, preservando el umbral económico de L1. En la práctica, las devoluciones "superan" al consenso como fuente de verdad efectiva para el usuario porque se avalan por pruebas criptográficas y colateral—no por reorgs—, reforzando así la seguridad y la experiencia de pago.

Apéndice: Especificación resumida (pseudo)

Inputs: h, amount a, policy P, vk, committee pubkey, collateral c

1. Commit(P), lock(c) on L1
2. R := {h, a, meta}
3. π := Prove_zk(P, R)
4. σ := Sign_BLS(committee, R)
5. S' := T(S, R); root' := MerkleRoot(S')
6. Anchor(root') on L1
7. ρ := MerkleProof(S', receipt_for(h,a))

Verifier:

accept $\Leftarrow$ Verify_zk(vk, π) $\wedge$ Verify_BLS(σ) $\wedge$ VerifyMerkle(root', ρ)