Budgeted Threshold Signatures: Protocol-Native Spending Limits with Zero On-Chain Footprint

Huaifeng Chen[1], Yuchang Zhang[1], and Yu Cheng[2]

[1] North China Institute of Computer Systems Engineering
chenhf@ncse.com.cn, zhangyuchang26@mails.ucas.ac.cn
[2] Beijing Electronic Science and Technology Institute
chengyu_cs@zjut.edu.cn

Abstract. Spending limits are the workhorse of custody policy, yet in every deployed threshold-signing stack they live outside the signing protocol—in centralized policy engines, on-chain covenants, or hardware co-signers—where they leak budget state, bind to a single chain, or add a trusted bypass surface. The workload is compounding: agent payments passed one hundred million transfers on a single chain in under a year. We initiate the study of protocol-native spending limits: per-epoch budgets secret-shared among the signers and enforced inside the signing session, with unchanged on-chain verification. We first show that the natural approach fails: gating a threshold signature with a passively secure MPC comparison admits a deterministic bypass. For the class of *linear gate-in-combination* protocols we prove that every member signs over-budget transactions with probability one, disguised against any transcript auditor up to the statistical masking bound. We then construct BTS, a budgeted threshold Schnorr scheme with a formal budget-soundness definition for honest-majority configurations. The gap is closed with an authenticated gate: in implementation, every attack attempt is caught before a signature is produced, at a measured cost of 3.4–3.7× multiplications and five extra rounds. Batched over k=20 payments, enforcement amortizes to about one network round trip per payment in our WAN-modeled prototype.

Keywords: Threshold signatures · MPC · Spending limits · Custody wallets

1 Introduction

Threshold custody—a t-of-n quorum of signers jointly controlling a single on-chain address—is the dominant architecture for institutional digital-asset wallets and is increasingly common for AI-agent payment wallets. Every serious deployment of this architecture ships with *spending limits*: per-period maximal amounts that the wallet may transfer, enforced through time-aggregated rules.

The deployment record is consistent and explicit: Fireblocks' Policy Engine exposes time-aggregated transaction limits (`TIMEFRAME` amount scopes accumulating over a period) evaluated in a centralized workflow layer *before* MPC signing is attempted [21]; its x402 Agent Spending Controls layer markets "budgets, per-agent isolation, [and] Policy-enforced limits" explicitly as features of that external engine [22]; BitGo ships velocity limits as custody-service policies which, by BitGo's own documentation, are bypassed by recovery flows that use the user and backup keys [5]; Nunchuk enforces limits through a platform co-signing key [38]; and Google's AP2 mandates carry `budget` and `amount_range` constraint fields whose enforcement is delegated to outer protocol participants [20]. The workload they guard is also compounding: agent-initiated payments already settle at a median between $0.01 and $0.10 per transaction (176 million payments, 98.6% in USDC [27]) and have passed one hundred million cumulative x402 transfers on Base within a year [10]—a high-frequency, small-ticket regime in which per-transaction overheads dominate.

Across all of these mechanisms, the limit lives *outside* the signing protocol. The consequences are structural rather than incidental. First, *budget privacy*: a centralized engine or public contract sees the budget policy and, in the contract case, the running expenditure. Second, *chain binding*: on-chain limit covenants [36,28] are per-chain constructions that change the address format and reintroduce gas costs. Third, and most sharply, a *bypass surface outside the cryptographic core*: the components that enforce the limit are exactly the components an adversary aims to subvert, and the vendors' own documentation delineates these exemptions. What no deployed stack offers is the property one would naturally want from threshold custody: a signature over an over-budget payment that *cannot be produced by any compliant signing quorum*, with the budget state held under the same secret sharing that already protects the key.

We study that property. Our finding is that obtaining it is subtle: the obvious construction—gate a threshold Schnorr signature on a passively secure comparison of the remaining budget—is deterministically bypassable. We prove this for a general class of protocols, then close the gap with an authenticated gate and measure the price of integrity end to end.

A separation result. We define *linear gate-in-combination* protocols (class $\mathcal{L}$; Sect. 4): the budget verdict is computed on field-shared values whose adversary-reachable influence is an affine injection (the regime of passively secure MPC "secure up to additive attacks" [23,6]), the signature is aggregated by group-linear combination as in FROST [14], and the gate path carries no binding mechanism. For this class we prove:

Theorem 1 (informal; deterministic bypass). *For every protocol in $\mathcal{L}$ and every single corrupted signer, there is a deterministic polynomial-time adversary that makes the protocol output a valid signature on any over-budget payment with probability* 1*, while the honest parties' joint view is statistically indistinguishable, to within the $2^{-\kappa}$ masking bound, from an honest execution on the same message under a shifted budget $B' \geq a$—disguise against any auditor that sees only the protocol transcript, including commitments from verifiable secret sharing.*

The attack behind Theorem 1 is constructive and we implement it: a corrupted signer biases one resharing in the verdict computation by an offset derived from public Lagrange coefficients, flipping every rejected payment to acceptance with zero detectability (Sect. 4; 100% false acceptance across four custody configurations). Two natural patches fail in instructive, measured ways: random biases merely destroy availability, and Feldman VSS [19] detects inconsistent *sharings* (our split-brain experiments: 100% detection with attribution) but not biased *values*, because it authenticates the sharing mechanism rather than the value carried.

The construction. BTS (Sect. 5) keeps FROST2 signing untouched—on-chain verification and key formats are unchanged, so the wallet remains a standard Schnorr address—and maintains the epoch budget as a Shamir-shared state among the n signers. Each payment runs a budget comparison over the shared state; an honest signer contributes its signing share only after a passing, verified check. The gate itself is *authenticated*: applying circuit randomization in the style of Chida et al. [12], every wire carries an information-theoretic tag under a session-global, never-opened MAC key, and a Fiat–Shamir random linear combination over all wires catches any additive bias with probability $1 - O(T/|\mathbb{F}|)$. The matching upper bound completes the separation:

Theorem 2 (informal; authenticated gate). *The authenticated gate renders every additive bias of the comparison circuit detectable before any signing round, except with probability $O(T/|\mathbb{F}|)$, where T is the circuit's wire count (loosely, by a union bound; the per-session bound is $3/|\mathbb{F}|$, Theorem 4).*

In implementation the attack succeeds in every trial against the unauthenticated baseline and is caught in all 800 trials of the authenticated gate, across four configurations—the alarm fires inside the budget check, so no signature is produced; randomizing the attack buys the adversary nothing (1,200 further trials, EXP-017). The measured price is 3.4–3.7× circuit multiplications and five extra rounds; a Feldman layer adds split-brain attribution.

Amortization. Machine-payment scale makes serial gating binding: a stack that checks every payment one at a time tops out below one payment per second, while batched enforcement sustains 4–6 payments per second. Naïve per-payment gating is inherently serial: the $(i{+}1)$-st comparison depends on the updated state, so pipelining is unavailable by protocol structure. Batch mode (Sect. 6) exploits a simple lemma: amounts are non-negative, so every prefix sum is bounded by the total, and a single comparison of the batch total therefore covers a k-payment batch. With $k{=}20$ and the authenticated gate deployed, enforcement amortizes to about one network round trip per payment; end-to-end, budgeted payments complete in 163–244 ms each over a 100 ms-RTT WAN—1.1–1.4× faster than signing the same payments one at a time, the serial regime of current custody deployments. The envelope is measured, not assumed: batched enforcement beats serial signing above a per-quorum bandwidth floor of 1.4–87 Mbps (Sect. 7).

Contributions.

1. **Definitions.** To our knowledge, the first formal treatment of protocol-native spending limits: budget soundness, budget privacy, fail-closed liveness, and threshold unforgeability for budgeted threshold signing (Sect. 3).
2. **Lower bound.** The class $\mathcal{L}$ and Theorem 3: deterministic bypass of every unauthenticated linear gate-in-combination protocol, statistically disguised to the masking bound, instantiated and measured (Sect. 4, 4).
3. **Upper bound.** BTS with an authenticated gate; Theorem 4 and its empirical closure (attack success 100% $\to$ 0 at measured cost), with Feldman attribution as an orthogonal layer (Sect. 5).
4. **Amortization.** The prefix-sum equivalence lemma and batch mode: with the authenticated gate, enforcement at 1.30–1.40 round trips and 1.16–1.43$\times$ compute per payment, with a measured envelope: batch-size curve to 0.27 round trips at k=100 and bandwidth floor 1.4–87 Mbps (Sect. 6, 7).
5. **Prototype and unified accounting.** A pure-stdlib Python prototype and a cost accounting (rounds, traffic, measured scalar multiplications, WAN model, multi-seed variance) that separates structural ratios from prototype-scale wall-clock (Sect. 7).

2 Related Work

Threshold signing and custody. FROST [32] and FROST2 [14], standardized as RFC 9591 [13], are the basis of our signing layer; two-round multi-signature pitfalls [18] motivate our session binding. Among recent custody constructions, BLISK compiles *boolean* authorization policies into a single verification key [33] but expresses no quantitative budget and keeps policy state public; Iceberg nests threshold groups inside MuSig2 channels for Lightning [24] without gating; Porechna composes ordinary signatures with Shamir-sealed authorization [40] as a concept proposal without budget state, attacks, or implementation. None of these addresses spending limits inside the signing protocol. Accountable threshold signatures with proactive refresh [7] are the closest template of a custody-motivated new property added to threshold signing; our property concerns quantitative policy enforcement rather than accountability. Threshold deployments that move limit checking to consensus-level scripts inherit the chain-binding and privacy costs that motivate protocol-native enforcement, and the latency price of threshold cryptosystems in blockchains has itself been measured [43].

MPC rate limiting and policy. Rate-limiting nullifiers and their collaborative variants [25] enforce messaging-speed constraints for privacy pools, not signing authorization. Zero-knowledge regulated payments (e.g., [36] for the covenant line and policy-compliant signatures generally) move compliance into proofs attached to transactions; our concern is the complementary point in the stack—the moment a quorum of signers must refuse an over-budget transaction.

Secure comparison. Constant-round comparison in honest-majority MPC goes back to Bar-Ilan–Beaver unbounded fan-in multiplication [1] and DFKNT [16], with practical primitives by Catrina–de Hoogh [9] and recent constant-round

protocols [35,37]; the PPML line instantiates comparison-gated decision paths at scale [46,45]. Our own comparison sub-protocol is textbook (a masked opening with a logarithmic-depth borrow-prefix tree), treated as a replaceable component; Sect. 7 tabulates the rounds/communication trade-off against the constant-round line.

System-layer proposals. 0xPass [31] and provenance-scoped agent wallets [44] place authorization in an orchestration layer above blockchains; delegated Bitcoin payments [42] formalize fair exchange for agent commerce. Guardian role-gates agent wallets by releasing a governor share only after external policy checks [39]—the same pattern. These proposals are complementary: they presume an enforcement point that this paper supplies inside the signing protocol.

Additive attacks and authentication. That passively secure MPC is "secure up to additive attacks" is the Genkin–Ishai line [23], recently extended to opening gates [6]. SPDZ-style MACs [17], BDOZ [4], MASCOT [29], Overdrive [30], and honest-majority authenticated protocols [12,34] supply the countermeasures; identification-focused variants [11,15,26] and the delayed-consistency-check vulnerabilities of Brüggemann–Schneider [8] inform our design choices (joint single-bit checks; no late pairwise comparisons); Sect. 4 discusses what additive errors mean when the gate carries money.

3 Preliminaries and Model

Signing layer. Parties $P_1, \ldots, P_n$ hold Shamir shares s_i of a Schnorr secret s over $\mathbb{Z}_N$ (N the order of the secp256k1 base point G); the group key is $Y = sG$. Key shares may be produced by a FROST DKG or by a trusted dealer (the wallet owner); our prototype uses dealer keygen, and the budget state is dealt by the same owner—both are orthogonal to the gate. FROST2 signing on message m proceeds in two rounds (nonce commitments with per-signer binding factors; partial signatures aggregated to σ with R, eY-verification). We treat FROST2 as a black box satisfying EUF-CMA under $\leq t-1$ corruptions and require *no change* to its transcript or to on-chain verification.

Budget layer. The epoch budget $B_0 < 2^\ell$ (ℓ=64) is shared as a degree-f Shamir sharing $[B]$ over $\mathbb{F} = \mathbb{Z}_N$ among the same n parties, who evaluate it with BGW-style arithmetic [3,41]: additions are local; multiplication uses local products reshared at degree f with Lagrange-at-zero recomposition. Amounts a are public (transaction semantics). The comparison sub-protocol (Sect. 5) opens a masked value $w = c + r$ with r uniform over $\ell + \kappa + 1$ bits (κ=40, the MP-SPDZ default statistical security), so w leaks nothing about c beyond $2^{-\kappa}$; a single verdict bit is opened.

Threat model. A static PPT adversary corrupts at most $t-1$ parties for the signing layer (FROST2's threshold) and at most $f \leq t-1$ for the budget layer with $n > 2f$ (honest majority). Table 1 shows that the custody-standard configurations 2-of-3, 3-of-5, 4-of-7, and 5-of-9 all satisfy both constraints simultaneously. The n=t regime (e.g., 2-of-2) falls outside honest-majority BGW and is discussed

Table 1. Two-layer corruption compatibility for custody-standard configurations: signing tolerates $t-1$ corruptions (FROST2), budget layer requires $n > 2f$ with $f = t-1$.

Config	$t-1$	$n > 2(t-1)$	Budget layer	BTS
2-of-3	1	$3 > 2$	✓	✓
3-of-5	2	$5 > 4$	✓	✓
4-of-7	3	$7 > 6$	✓	✓
5-of-9	4	$9 > 8$	✓	✓
2-of-2	1	$2 > 2$	dishonest majority (Sect. 8)	

in Sect. 8. The baseline budget layer is passively secure; Sect. 5 adds authentication for active adversaries on the gate path (detect-and-abort; Sect. 4 shows why). Communication is authenticated point-to-point with broadcast; delivery is eventual.

Security definitions. Let Verify be standard Schnorr verification and $\mathsf{amt}(m)$ the amount carried by m.

Definition 1 (Threshold unforgeability). *No PPT adversary corrupting $\leq t-1$ parties, given the full protocol view including budget-layer transcripts, produces (m, σ) with $\mathsf{Verify}(\sigma, m) = 1$ unless an honest quorum signed m.*

Definition 2 (Budget soundness, per epoch). *Fix an epoch with budget B_0. Consider the game in which a PPT adversary corrupts $\leq t-1$ parties (and $\leq f$ on the budget layer in the authenticated variant) and adaptively schedules signing sessions on messages of its choice; honest parties follow the protocol, and each session either completes (all honest participants sign) or aborts. The adversary wins if it outputs $\{(m_j, \sigma_j)\}_j$, all valid under Verify and all carrying the epoch's state-binding tag, with $\sum_j \mathsf{amt}(m_j) > B_0$. Budget soundness holds if the winning probability is negligible.*

Definition 3 (Budget privacy). *The joint view of any $\leq f$ parties is statistically close to independent of B_0 and of intermediate balances, leaking at most the $2^{-\kappa}$ masking bound of opened values.*

Definition 4 (Fail-closed liveness). *If a payment passes its budget check and the required quorum is online, honest parties complete the signature; aborts occur only on detected integrity violations.*

4 The Naive Class and a No-Go Theorem

The natural construction—"run a passively secure comparison of $[B] - a$ against zero and sign if it passes"—fails in a strong sense. We characterize the class of protocols for which this is unavoidable.

Definition 5 (Linear gate-in-combination protocols, $\mathcal{L}$). *A budget-gated threshold Schnorr protocol Π belongs to $\mathcal{L}$ if:*

- ***(L0)*** Privacy premise*: no plaintext intermediate value of the balance (e.g., $B - a$) is revealed before the verdict;*
- ***(L1)*** Affine verdict path*: the verdict bit v is a deterministic function of a field-linear recombination $w = \sum_i \lambda_i g_i$ of per-party values g_i, each linear in that party's budget share and resharing randomness, and the adversary-reachable effect on w is an affine injection (the gate subcircuit is secure up to additive attacks [23,6]);*
- ***(L2)*** Group-linear combination*: partial signatures and the aggregate nonce are linear combinations ($\sigma = \sum_i \lambda_i \sigma_i$, $R = \sum_i R_i$) and validity is exactly the Schnorr equation $\sigma G = R + eY$;*
- ***(L3)*** No binding on the gate path*: the values $\{g_i\}$ and the verdict-path intermediates carry no binding mechanism—no information-theoretic or computational MACs (SPDZ, BDOZ, Rabin–Ben-Or information checking), and no randomized redundancy (algebraic manipulation detection, circuit randomization).*

L0 excludes protocols that simply open $B - a$ in the clear (which sacrifice Def. 3 and are caught by share-consistency checks); it is also what creates the attack surface: privacy forces a masked comparison, a masked comparison forces multiplication gates, and that is where affine influence enters. L1 covers the standard masked-comparison structure—the verdict is a function of an opened masked value w and shared mask bits—because BGW resharing randomness does not authenticate values: a corrupted party's resharing of its local product can carry an additive offset while remaining a perfectly consistent sharing [23]. L2 is the FROST-style aggregation. L3's exclusion list is drawn from the standard mechanism families; it is not tailored to our fix.

Theorem 3 (Deterministic bypass). *For every $\Pi \in \mathcal{L}$ and every single corrupted signer P_j, there is a deterministic polynomial-time adversary $\mathcal{A}$ such that for every over-budget message m ($\mathsf{amt}(m) = a > B$): Π outputs a valid signature on m with probability 1; and the honest parties' joint view is statistically indistinguishable, to within the $2^{-\kappa}$ masking bound, from an honest execution on the same m under a shifted budget $B' \geq a$.*

Proof sketch (full proof in Appendix A). *(1) Free verdict biasing.* By L1 the corrupted party shifts its contribution g_j by Δ/λ_j for any field Δ of its choice; the offset passes through the resharing as a consistent sharing and, crucially, is independent of the honest parties' randomness, so a masked bit-decomposition comparison admits a *masking lemma*: the Δ that flips the verdict is a function of public parameters only (public Lagrange coefficients and bit lengths). In our instantiation the final verdict XOR has verdict $v = t \oplus \beta$ for a public-linear term t and a shared bit β produced by a multiplication gate; biasing that single gate by $\delta = -(2\lambda_j)^{-1}$ maps $v \mapsto v + 1$, turning every *deny* into *accept* deterministically. *(2) Statistical disguise.* By L3 the corrupted party's outgoing messages are information-theoretically consistent with an honest local state (the map from local randomness to outgoing messages is surjective), so any transcript-only check—including Feldman VSS commitments over the sharings, which authenticate the sharing mechanism rather than the carried value—has exactly

zero advantage at the share level and none beyond the $2^{-\kappa}$ masking bound of the opened value. The auditor model is anyone holding the protocol transcript and on-chain data, including computationally unbounded auditors (Appendix A delimits the boundary). □

Remark 1 (Why the group layer does not help the attacker). A natural guess is to compensate a verdict bias with a bias on the group side. Neither unilateral form works. Shifting (σ_j, R_j) to $(\sigma_j + \delta', R_j + \delta' G)$ preserves per-signer verification $z_j G = R_j + e\lambda_j Y_j$ but shifts the aggregate equation $\sigma G = R + eY$ by $(1 - \lambda_j)\delta'$; conversely $(\sigma_j + \delta, R_j + \lambda_j \delta G)$ preserves the aggregate but breaks per-signer verification by $(1 - \lambda_j)\delta$. With the two group checks in place, every unilateral compensation fails on one of them, so the attack must live entirely on the field side; a fix must authenticate the gate path, not the signature path. The experiments in Sect. 4 confirm the division.

Theorem 3 is a semantic corollary of the additive-attack program: prior work treats additive errors as a benign intermediate state to be compiled away [23]; in budget-gated signing, the additive error *is* the semantics—an offset of δ crosses a monetary threshold—and the group side of the composition converts it into a publicly verifiable artifact.

5 Authenticated Gating: The BTS Construction

BTS keeps the FROST2 signing layer untouched and gates it with an authenticated budget comparison. The state $[B]$ is Shamir-shared among all n parties; each payment carries amount a and message m; the wallet binds the session to the state via $\mathsf{tag} = H(\mathsf{epoch} \| \mathsf{seq})$ folded into m, so stale-state replays yield different signed messages and are flagged by any state-log verifier.

Figure 1 depicts one session end to end.

BudgetCheck (authenticated comparison). Write $d = [B] - a$ and $c = d + 2^{\ell} \in [0, 2^{\ell+1})$; then $d \geq 0$ iff bit ℓ of c is 1. The parties open $w = c + r$ for a fresh shared random r uniform over $\ell + \kappa + 1$ bits, compute the borrow recurrence $\beta_j = a_j \vee (b_j \beta_{j-1})$ with $a_j = (1 - w_j) r_j$, $b_j = (1 - w_j) \vee r_j$ over a logarithmic-depth semiring tree, and open the single verdict bit. This textbook core [9] satisfies L0–L2; what closes Theorem 3 is the authentication wrapped around it.

Circuit randomization (the authenticated gate). Following the honest-majority technique of Chida et al. [12], chosen over sacrifice-based triples [34,2] because it authenticates opened values without preprocessing infrastructure, every wire x of the comparison circuit carries a shared tag $[t_x]$ with invariant $t_x = \gamma x$ for a session-global key $[\gamma]$ that is generated once and never opened. Linear gates are local; each multiplication gate costs one second multiplication on shared values, $[t_{xy}] = [t_x] \cdot [y]$; all per-party random-bit inputs are certified ($[\gamma][b_i]$); every multiplication output and certified input is registered. Before the verdict is accepted, the parties run a single batched check: a fresh random value r^* is opened after all circuit wires are computed and folded into the

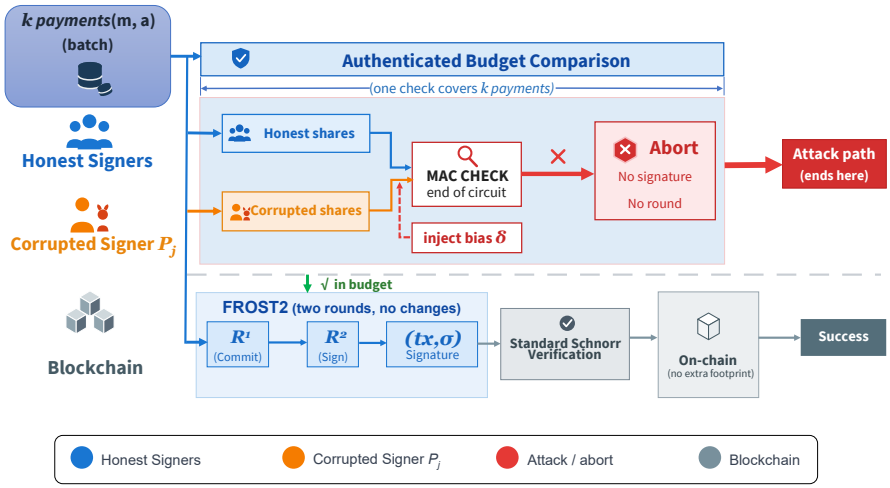

Fig. 1. Execution flow of BTS against a malicious signer. The signers jointly evaluate one authenticated budget comparison over the shared batch state; any bias δ injected by the corrupted signer is caught by the end-of-circuit MAC check, aborting the session before any signing round—no signature is ever produced. On the in-budget path, the standard two-round FROST2 signing (commit, sign) executes unchanged, and (tx, σ) is verified on-chain as a perfectly standard Schnorr signature, leaving zero on-chain footprint.

Fiat–Shamir seed (every gate bias is thus committed before the challenge coefficients are fixed), coefficients ρ are derived over the canonical transcript, and $T = [\gamma]\Big(\sum_j \rho_j[z_j]\Big) - \sum_j \rho_j[t_{z_j}]$ is verified to be zero via $Q = [\eta]T$ for a fresh shared η, opened once. A non-zero opening aborts the session before any signing round.

Theorem 4 (Authenticated gate). *In the authenticated gate, every additive bias of any multiplication gate or certified input of the comparison circuit leads to an abort (before any signing round) except with probability at most $3/|\mathbb{F}|$ per session; the abort leaks no information about honest inputs beyond opened values. With coin-flip challenges this bound is information-theoretic; our SHA-256 Fiat–Shamir instantiation realizes it in the random-oracle model.*

Proof sketch. A bias d on a gate output keeps the invariant only if the paired tag shifts by γd; with γ uniformly random and never opened, the adversary's tag bias is independent of γd and the aggregated check $T \neq 0$ survives the Fiat–Shamir coefficients with probability $\leq 2/|\mathbb{F}|$ (Chida et al., Lemma 4.2 lineage [12]); the η-randomized zero check adds $1/|\mathbb{F}|$ and leaks nothing on failure. The end-of-circuit placement covers *every* gate and input, so early biases cannot escape; the pre-opened r^* removes adaptivity of the challenge. □

Table 2 summarizes the empirical closure: across 800 attack trials the attack success rate is 0 and detection is 100%, in both the value-only and dual-bias

Table 2. Attack surface under BTS's authenticated gate. Attack success = fraction of denied payments signed; detection = fraction of sessions aborted before any signing round. Final-XOR is the deterministic bypass of Theorem 3 (bias $\delta = -(2\lambda_j)^{-1}$ at the verdict gate); dual = value and tag both biased. 100 trials per configuration and attack mode; clean runs 100/40/15/8 for 2-of-3 through 5-of-9.

	Final-XOR (value-only)		Final-XOR (dual)		Clean runs	
Config	Success	Detected	Success	Detected	False aborts	Wrong verdicts
2-of-3	0.00	1.00	0.00	1.00	0/100	0
3-of-5	0.00	1.00	0.00	1.00	0/40	0
4-of-7	0.00	1.00	0.00	1.00	0/15	0
5-of-9	0.00	1.00	0.00	1.00	0/8	0

modes; a further 20-trial negative control confirms that *should-pass* payments under attack are also detected (20/20) rather than silently denied—detection is independent of the verdict direction. At the wallet level, an attacked session raises the alarm inside BudgetCheck, before any FROST round, so no signature is ever produced (enforced by the implementation's call order and verified in tests). The price of integrity: the authenticated circuit costs 3.4–3.7× the multiplications of the (shortcut-optimized) baseline and five extra rounds.

Feldman attribution (orthogonal layer). A Feldman-VSS layer over the resharings detects *inconsistent sends* (split-brain): in our experiments, 50/50 (2-of-3) and 6/6 (3-of-5) split-brain attacks are detected and attributed to the misbehaving party, whereas the same attacks under the plain layer abort with the blame falling on an arbitrary (often honest) party. Feldman does not intercept value biases—its commitments authenticate the polynomial, not the carried value. The verification cost is structural: $n(n+1)(f+1) + n^2$ scalar multiplications per BGW multiplication, matching our instrumented measurements exactly (33/115/273 for 2-of-3/3-of-5/4-of-7; counts are deterministic per run, over 10/3/2 runs).

6 Batch Mode

Per-payment gating is inherently serial: the comparison for payment $i+1$ reads the state updated by payment i, so no pipelining of budget checks (nor of the interleaved signing sessions) is possible without weakening the invariant. Batch mode removes the serialization.

Lemma 1 (Prefix-sum equivalence). *Amounts are non-negative; hence for any batch $\{a_1, \ldots, a_k\}$ with $\sum_i a_i \le B$, every prefix sum satisfies $\sum_{i \le j} a_i \le B$. A single comparison of $[B] - \sum_i a_i$ against zero therefore accepts the batch iff all sequential per-payment comparisons would accept, and the post-batch balance equals the sequential outcome (verified on 40 randomized payment streams in our prototype).*

One comparison covers the whole batch; the k FROST sessions then run concurrently (their nonce commitments and partial signatures are batched into shared rounds). The all-or-nothing semantics (a single over-budget payment rejects the batch) is handled by bisection re-submission: with m bad payments in a batch of k, the expected number of additional comparisons is $O(m \log(k/m))$; for $m \ll k$ this stays within a constant factor of one comparison per payment. Measured on synthetic agent-payment-scale streams (lognormal amounts around the \$0.01–\$10 ticket range): accepted batches cost exactly one comparison, a rejected batch isolates the offending prefix in a median 4 (max 5) bisection comparisons at k=20, matching $\log_2 k$, with zero false prefix rejections (EXP-019).

Measured on our prototype (Sect. 7) with the authenticated gate: batch k=20 replaces one authenticated comparison per payment with one per batch, an end-to-end speedup of 13.9–16.7× per payment relative to sequential authenticated gating; end-to-end over a 100 ms-RTT WAN, budgeted payments complete in 163–244 ms each versus 227–276 ms for unguarded *serial* per-payment signing (the regime of current custody deployments, in which each payment waits its own round trips) and versus 2.7–3.4 s for unbatched authenticated per-payment gating. Against the symmetric pipelined baseline—k unguarded FROST sessions run concurrently—batched BTS costs 2.8–4.4× per payment; we report the symmetric number alongside the serial one because serial baselines flatter any batching scheme, ours included. Sweeping the batch size at 3-of-5 ($k \in \{5, 10, 20, 50, 100\}$, EXP-015): the amortized network cost falls monotonically to 0.27 round trips per payment at k=100 while the compute ratio converges to 1.08×; k=20 sits on the steep part of the curve, so larger batches keep improving enforcement.

7 Evaluation

Prototype. A single-process, pure-stdlib Python simulator (secp256k1 with Jacobian coordinates, identity-safe group API; FROST2 with per-share verification; Shamir/BGW with degree-reduction and batched-round accounting; the authenticated gate with SHA-256 Fiat–Shamir challenges over canonical transcripts; attack hooks with trigger counting). Cost-structure figures are medians over multi-seed runs ($\geq$ 3 seeds, EXP-012); the WAN accounting of Table 3 uses medians over 15 fresh iterations of the authenticated pipeline (EXP-007c and EXP-013); independent reruns confirm the split: rounds, per-check traffic, and multiplication counts are identical across runs, while shared-machine wall-clock medians vary by 9–85% (EXP-021)—which is why we lead with structural quantities. Modules pass self-tests including boundary and randomized comparison cases, malformed-share detection, ledger semantics, and MAC tag-algebra regression against the (test-only opened) session key.

Cost structure of the budget check. The baseline (shortcut-optimized) comparison runs in 18–20 broadcast rounds, 143 KiB–2.8 MiB per check (2-of-3 through 5-of-9), with rounds scaling as $\log_2 \ell$ and traffic linearly in $\ell + \kappa$

(sweeps at $\ell \in \{32, 48, 64\}$, $\kappa \in \{20, 40\}$). With the offline/online split (shared-bit generation and mask folding precomputed), the *online* phase is 13–15 rounds and 27–74 KiB. We use κ=40 (the MP-SPDZ default) throughout. Against published protocols, honest-majority constant-round comparison achieves 3–4 online rounds (LTZC1 [9], P3R [35])[3], with 2-round plain-model results at the frontier [37]; the Bar-Ilan–Beaver/PreOrC line composes directly with our gate and would cut the online rounds to 3–6 at higher per-round bandwidth. We implement the textbook tree. As a measured step along this axis, a fused single-wave composition of our own tree, rewriting the borrow recursion in affine form $f_j(x) = A_j + B_j x$ so that both products per level batch into one communication wave, cuts the online phase from 13–15 to 8–9 rounds at +9–14% traffic and +2–4% multiplications (EXP-016; equivalence with the two-wave tree verified on 201 boundary and random cases): the wave count, not the multiplication count, is what the dependency chain between the AND and OR waves was costing. The cut reaches the authenticated gate as a strict improvement: the MAC-checked fused comparison measures 17–19 rounds against 22–25 for the two-wave tree, at *lower* traffic (−7–15%) and fewer multiplications (−7–14%: the OR of the two-wave compose costs a third tagged product per node), with equivalence verified on 171 boundary and random cases (EXP-020). The remaining gap to the 3-round line is unbounded fan-in machinery, which in the information-theoretic setting requires $n > 3f$ (not met by any custody configuration), leaving the computational or heavier-preprocessing routes. Our opening pattern (one masked value and one verdict bit) does not instantiate the prefix-OR opening mode whose leakage was recently exposed in the PPML line [46].

End-to-end WAN accounting. Figure 2 visualizes the five modes; Table 3 reports the symmetric five-row accounting at 100 ms RTT with k=20, measured with the *authenticated* gate throughout (2-of-3 through 5-of-9), under a latency-bound premise of ≥100 Mbps links—the authenticated gate raises per-round traffic to 13–293 KiB, which becomes material on 10 Mbps links: unguarded signing serial and pipelined; authenticated gating serial; batched BTS pipelined and its serial upper bound. The headline structural ratios: amortized network cost 1.30–1.40 round trips per payment; compute overhead ratio 1.16–1.43×; per-payment latency 163–244 ms, i.e., 1.1–1.4× faster than serial per-payment signing. The unbatched authenticated column (2.7–3.4 s) quantifies what batching buys; the pipelined-unguarded column quantifies what integrity costs. Measured bandwidth crossovers (EXP-014): batched BTS beats serial unguarded signing above $B^* \in \{1.4, 8.8, 29, 87\}$ Mbps for 2-of-3 through 5-of-9 (check traffic measured, FROST traffic analytic from RFC 9591 message sizes); the offline/online split bounds the online check at 27–74 KiB, so at 10 Mbps the serialized online traffic amortizes to a few milliseconds per payment.

[3] Round counts follow each source's own accounting: LTZC1 reports 3 online rounds and P3R 3–4, both in the preprocessing model over broadcast. Our 18–20 is the total inline count including shared-bit generation; the split above gives the 13–15 online figure.

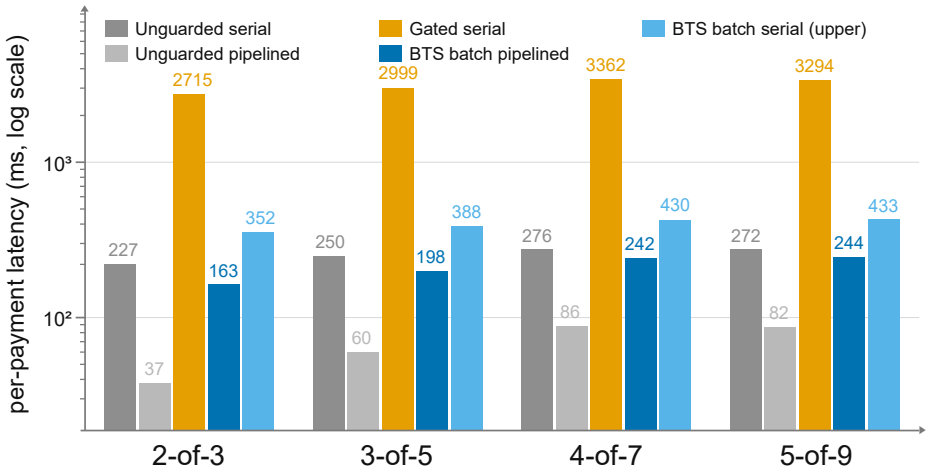

Fig. 2. Per-payment latency over a 100 ms-RTT WAN (k=20, authenticated gate, prototype compute medians). Batched BTS (blue) sits below the serial-signing regime of current deployments across configurations; the log scale spans unbatched authenticated gating (two orders of magnitude) down to pipelined unguarded signing.

Authenticated-gate and verification costs. The authenticated gate costs 3.4–3.7× circuit multiplications and +5 rounds (consistent accounting +6) over the optimized baseline; its multiplication count matches the structural closed form: one input-certification round over $105n + 1$ certified inputs (the n parties' mask bits plus the budget state), a doubled circuit (each gate, including the final verdict XOR, carries its tag product), and two finalize multiplications—measured 1118 and 1748 multiplications for 2-of-3 and 3-of-5. The Feldman layer's verification cost is $n(n{+}1)(f{+}1) + n^2$ scalar multiplications per BGW multiplication, confirmed by instrumentation (33/115/273 for 2-of-3/3-of-5/4-of-7).

Privacy. Across 10^4 simulation-level trials of the masking construction, the opened masked value correlates with the true comparison operand at -0.008 (Pearson), exact recovery succeeds 0 times (design bound 2^{-40}), and minority share sets are consistent with every candidate budget (zero information; Shamir perfect secrecy), confirming Def. 3 empirically.

Attack surface (summary). The unauthenticated baseline shows the motivating failures: the final-XOR bias flips 100% of denied payments undetected, random global biases merely destroy liveness, and split-brain sends abort with arbitrary blame. The authenticated gate answers the first (Table 2); Feldman attribution answers the third (Sect. 5).

8 Discussion and Limitations

Dishonest-majority configurations. Configurations with $n \leq 2f$ (e.g., 2-of-2) fall outside honest-majority BGW. The authenticated gate composes with

Table 3. Symmetric WAN accounting at 100 ms RTT, k=20 payments per batch (ms per payment), measured with the authenticated gate (EXP-007c; 5-of-9 from EXP-013). Net = amortized network round trips per payment.

	Unguarded FROST		Gated	BTS batch	
Config	serial	pipelined	serial	pipelined	serial (upper)
2-of-3	227	37	2715	**163**	352
3-of-5	250	60	2999	**198**	388
4-of-7	276	86	3362	**242**	430
5-of-9	272	82	3294	**244**	433
Net (RTT/pay)	2.0	0.1	26–27	**1.30–1.40**	

SPDZ-style preprocessing [17,30] at the cost of triple generation; we leave an implementation in that regime to future work. The custody-standard configurations of Table 1 all satisfy the honest-majority constraint.

Concurrency. Honest parties serialize budget-check sessions locally and sequence sessions by $(\mathsf{epoch}, \mathsf{seq})$; Appendix B states the linearization lemma: under these rules, any interleaving has an equivalent sequential execution, and budget soundness is preserved by induction over the linearization order. The prototype is a single-process simulator; message-level interleaving is engineering future work.

Liveness and crash recovery. Epoch checkpointing with quorum resharing (the same mechanism as proactive refresh [7]) restores a returning signer; DoS by abort is inherent to detect-and-abort at this corruption threshold (guaranteed output delivery requires $f < n/3$), and attribution (Feldman layer) is the operational answer.

Corruption timing. Our model is static, matching FROST2's analysis. The budget layer's detection is per-session and timing-agnostic: retries across 100 fresh sessions are detected 100/100 with no accumulated advantage (the session MAC key is never reused), and honest sessions after an abort complete cleanly (EXP-018).

Predicate generality. Nothing in the class definition or the authenticated gate is specific to budgets: any quantitative policy expressible as a comparison over shared state (per-window rate limits, velocity caps, portfolio concentration limits) inherits both the no-go result and the construction, with the amount semantics carried by the same state-binding tag. We instantiate spending limits because they are the deployed demand; the separation is a property of the gate, not of the currency semantics.

Deployment fit. The workload evidence, agent payments at a median below \$0.10 [27], matches the batch regime in which enforcement is cheapest. BTS requires no change to chain validation, wallets remain standard Schnorr addresses, and the amortized network cost (one round trip per payment) sits below the latency budget of custody operations. A full network deployment (real sockets,

distributed parties) is future engineering; our WAN accounting is an analytic model on measured per-phase costs, reported with symmetric baselines.

9 Conclusion

Spending limits belong inside the signing protocol that custody already runs. We showed that the natural way to put them there fails deterministically for a broad class of protocols, closed the gap with an authenticated gate whose measured cost is a small constant factor, and found that batching brings the amortized price of enforcement down to about one network round trip per payment. The construction is not specific to budgets—any quantitative policy over shared state inherits both the no-go and the gate. We hope the budget-soundness definitions prove useful wherever threshold protocols are asked to enforce policy.

A Full Proof of Theorem 3

Step 1 (free verdict biasing). Fix $\Pi \in \mathcal{L}$, corrupted party P_j, over-budget payment (m, a) with $a > B$. By (L1) the verdict is a deterministic function of $w = \sum_i \lambda_i g_i$ where g_j is P_j's contribution. For any field element Δ, replacing g_j by $g_j + \Delta/\lambda_j$ shifts w by Δ; in a BGW resharing this is realized by biasing the local product at the resharing of the target gate by $\lambda_j^{-1}\Delta$ against the Lagrange-at-zero recomposition, and the resulting sharing is internally consistent (its shares lie on a degree-f polynomial), hence indistinguishable from an honest resharing at the share level. Because the honest parties' resharing randomness is independent of Δ, the masked comparison admits a masking lemma: the verdict-relevant bits of the opened masked value w are public, and the offset that maps a deny verdict to accept is a function of public parameters alone. Concretely, for the bit-decomposition comparison of Sect. 5, the verdict is $v = w_\ell \oplus r_\ell \oplus \beta_{\ell-1}$ where $\beta_{\ell-1}$ is the output of the final borrow-prefix composition—an affine-composed function whose last multiplication gate output, biased by $\delta = -(2\lambda_j)^{-1}$, changes the shared verdict bit v to $v + 1$; every deny ($v = 0$) becomes accept ($v = 1$) with probability 1, independent of all randomness.

Step 2 (statistical disguise). By (L3) no binding mechanism exists on the gate path. The map from P_j's local randomness to its outgoing messages, conditioned on any fixed honest view, is surjective onto the set of messages consistent with the (biased) resharing: for linear secret sharing, every outgoing share vector of a valid degree-f sharing arises from some choice of local randomness. Hence the joint honest view under the attack is statistically indistinguishable, to within the $2^{-\kappa}$ masking bound of the opened value w, from the view under an honest execution with an adjusted local state (equivalently, the same m under a shifted budget); in particular, transcript predicates—including Feldman commitment checks, which verify polynomial consistency of sharings but not the values carried—have advantage negligibly above 0. The claim is information-theoretic and holds for

computationally unbounded transcript auditors. We make no claim about auditors holding external ground truth (e.g., the account owner's own records), which is outside the transcript model.

Step 3 (signature output). With the verdict flipped to accept, the honest parties proceed exactly as in an in-budget execution: each honest signer in the quorum contributes its FROST2 partial signature on m (the state binding tag uses the post-check sequence number, consistent with an accepted payment), and the aggregate is a valid Schnorr signature on the over-budget m. By (L2) nothing in the signature depends on the budget path beyond the boolean gate. □

The tightness corollary: removing (L3)—i.e., authenticating the gate path—yields the matching upper bound (Theorem 4); the separation is tight. Feldman-only protocols remain in $\mathcal{L}$ and are subject to Theorem 3, consistent with our split-brain/bypass experiments: VSS authenticates sharings, not values.

B Linearization Lemma

Lemma 2 (State linearization). *Suppose honest parties (i) participate in at most one budget-check session at a time (local serialization) and (ii) accept a sequence number only if it extends their local maximum (with ties resolved by public broadcast order, aborting on mismatch). Then for any interleaving of sessions there exists an equivalent sequential execution, and budget soundness (Def. 2) holds inductively over the linearization order.*

Proof. Sequence-number reservation under (i)–(ii) defines a total order on sessions that all honest parties agree on: two concurrent sessions cannot both reserve the same next sequence number at an honest party (local mutual exclusion), and the broadcast order resolves cross-party ambiguity (with abort on mismatch, which is fail-closed). Within each session, the honest partial signature is a function of a passing, authenticated verdict (the gate enforcement point), so each linearized session atomically either signs and deducts or aborts without deduction; the invariant $\sum$ signed amounts $\leq B_0$ is preserved across each atomic step by the comparison's correctness. An adversary holding $t-1$ shares cannot complete a signature outside a session (FROST2 unforgeability), and cannot pass an unauthenticated check in the authenticated variant (Theorem 4). □

C Why Not Compile the Gate with AMD?

AMD (algebraic-manipulation-detection) compilers [23] protect circuits statistically ($\varepsilon = O(|C|/|\mathbb{F}|)$, field-size conditions); our authenticated gate detects with per-session probability $1 - 3/|\mathbb{F}|$ in the honest-majority setting at comparable cost. More fundamentally, AMD protects the circuit's internal computation, but the binding between the field-side verdict and the group-side signature artifact sits outside any circuit; authenticating shared values is what ties the amount to the session.

D Reproducibility

The prototype is pure-stdlib Python 3; each module ships self-tests (`python3 -m src.btcrypto | src.budget | src.mac | src.bts`). Experiments are single-command scripts writing `metrics.json`: baseline and overhead sweeps (EXP-001 through 003), attacks (EXP-004 and 004b), privacy (EXP-005), batch (EXP-006), symmetric WAN accounting with the unauthenticated and authenticated gates (EXP-007b and 007c), the 5-of-9 completion (EXP-013), bandwidth sensitivity with measured crossovers (EXP-014), the batch-size sweep (EXP-015), the fused single-wave comparison tree (EXP-016), randomized-bias and cross-epoch attacks (EXP-017/018), synthetic-stream rejection recovery (EXP-019), the authenticated fused tree (EXP-020), WAN-table rerun variance (EXP-021), parameter sweeps and offline/online split (EXP-008 and 009), MAC closure (EXP-010), structural costs with instrumented scalar-multiplication counts (EXP-011 and 011b), and multi-seed statistics (EXP-012 and 012b). The anonymized artifact bundle accompanies the submission.

References

1. Bar-Ilan, J., Beaver, D.: Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In: Proceedings of the 8th Annual ACM Symposium on Principles of Distributed Computing (PODC 1989). pp. 201–209 (1989)
2. Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Advances in Cryptology – CRYPTO '91. LNCS, vol. 576, pp. 420–432 (1991)
3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC 1988). pp. 1–10 (1988)
4. Bendlin, R., Damgård, I., Orlandi, C., Zakarias, S.: Semi-homomorphic encryption and multiparty computation. In: Advances in Cryptology – EUROCRYPT 2011. LNCS, vol. 6632, pp. 169–188 (2011), ePrint 2010/514
5. BitGo: Policies overview. BitGo developer documentation (2026), `https://developers.bitgo.com/docs/policies-overview/`, accessed 2026-09-02
6. Blanton, M., Murphy, D., Yuan, C.: Efficiently compiling secure computation protocols from passive to active security: Beyond arithmetic circuits. Proceedings on Privacy Enhancing Technologies **2024**(1), 74–97 (2024)
7. Boneh, D., Partap, A., Rotem, L.: Proactive refresh for accountable threshold signatures. In: Financial Cryptography and Data Security (FC 2024) (2024), ePrint 2022/1656
8. Brüggemann, A., Schneider, T.: When trying to catch cheaters breaks the MPC: Breaking and fixing delayed consistency checks in trident, fantastic four, SWIFT, and quad. In: Advances in Cryptology – EUROCRYPT 2026 (2026), ePrint 2026/234
9. Catrina, O., de Hoogh, S.: Improved primitives for secure multiparty integer computation. In: Security and Cryptography for Networks (SCN 2010). LNCS, vol. 6280, pp. 182–199 (2010)
10. Chainalysis: Inside x402: 100m agentic payments on base. Chainalysis blog (2026), `https://www.chainalysis.com/blog/x402-agentic-payments-adoption/`, accessed 2026-09-03

11. Chaudhari, H., Rachuri, R., Suresh, A.: Trident: Efficient 4PC framework for privacy preserving machine learning. In: 26th Annual Network and Distributed System Security Symposium (NDSS 2020) (2020), ePrint 2019/1315
12. Chida, K., Genkin, D., Hamada, K., Ikarashi, D., Kikuchi, R., Lindell, Y., Nof, A.: Fast large-scale honest-majority MPC for malicious adversaries. In: Advances in Cryptology – CRYPTO 2018 (2018), ePrint 2018/570
13. Connolly, D., Komlo, C.: FROST (Flexible Round-Optimized Schnorr Threshold) signature scheme, RFC 9591. IETF (2025), https://www.rfc-editor.org/rfc/rfc9591
14. Crites, E., Komlo, C., Maller, M.: How to prove Schnorr assuming Schnorr: Security of multi- and threshold signatures (2021), preprint, ePrint 2021/1375
15. Dalskov, A., Escudero, D., Keller, M.: Fantastic four: Honest-majority four-party secure computation with malicious security. In: 30th USENIX Security Symposium. pp. 2183–2200 (2021)
16. Damgård, I., Fitzi, M., Kiltz, E., Nielsen, J.B., Toft, T.: Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In: Theory of Cryptography (TCC 2006). LNCS, vol. 3876, pp. 285–304 (2006), ePrint 2005/066
17. Damgård, I., Pastro, V., Smart, N., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Advances in Cryptology – CRYPTO 2012. LNCS, vol. 7417, pp. 643–662 (2012), ePrint 2011/535
18. Drijvers, M., Edalatnejad, K., Ford, B., Kiltz, E., Loss, J., Neven, G., Stepanovs, I.: On the security of two-round multi-signatures. In: 2019 IEEE Symposium on Security and Privacy (S&P 2019). pp. 1084–1101 (2019), ePrint 2018/417
19. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: 28th Annual Symposium on Foundations of Computer Science (FOCS 1987). pp. 427–437 (1987)
20. FIDO Alliance: AP2 payment mandate specification. ap2-protocol.org (2026), https://ap2-protocol.org/ap2/payment_mandate/, accessed 2026-09-02
21. Fireblocks: Configure transaction authorization policies. Fireblocks API reference (2026), https://developers.fireblocks.com/reference/configure-transaction-authorization-policy, accessed 2026-09-02
22. Fireblocks: x402 agent spending controls. Fireblocks developer documentation (2026), https://developers.fireblocks.com/docs/x402-agent-spending-controls, accessed 2026-09-02
23. Genkin, D., Ishai, Y., Prabhakaran, M., Sahai, A., Tromer, E.: Circuits resilient to additive attacks with applications to secure computation. In: Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 2014). pp. 495–504 (2014), ePrint 2015/154
24. Gerhart, P., Kohen, N., Posner, J., Furszyfer, M.: Enabling threshold custody for the lightning network with nested threshold multi-signatures (2026), arXiv 2608.20705
25. Gürel, Y., Şen, U., Yayla, O.: Collaborative rate limiting nullifier signaling (2026), ePrint 2026/1259
26. Harth-Kitzerow, C., Suresh, A., Wang, Y., Yalame, H., Carle, G., Annavaram, M.: High-throughput secure multiparty computation with an honest majority in various network settings. Proceedings on Privacy Enhancing Technologies **2025**(1), 250–272 (2025), ePrint 2024/386
27. Harvey, B.: Who pays the agent? the race for frictionless machine payments. Keyrock research report (2026), https://keyrock.com/who-pays-the-agent/, accessed 2026-09-02

28. Heilman, E., Kolobov, V.I., Levy, A.M., Poelstra, A.: ColliderScript: Covenants in Bitcoin via 160-bit hash collisions (2024), ePrint 2024/1802
29. Keller, M., Orsini, E., Scholl, P.: MASCOT: Faster malicious arithmetic secure computation with oblivious transfer. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS 2016). pp. 830–842 (2016), ePrint 2016/505
30. Keller, M., Pastro, V., Rotaru, D.: Overdrive: Making SPDZ great again. In: Advances in Cryptology – EUROCRYPT 2018. LNCS, vol. 10822, pp. 158–189 (2018), ePrint 2017/1230
31. Kim, K., Chelikavada, K.: 0xPass: A secure protocol for universal cross-chain accounts (2026), arXiv 2608.18359
32. Komlo, C., Goldberg, I.: FROST: Flexible round-optimized Schnorr threshold signatures. In: Selected Areas in Cryptography (SAC 2020). LNCS, vol. 12804, pp. 34–65 (2020), ePrint 2020/852
33. Kurbatov, O., Hrubiian, Y., Melnyk, I., Antadze, L.: BLISK: Boolean circuit logic integrated into the single key (2026), ePrint 2026/088
34. Lindell, Y., Nof, A.: A framework for constructing fast MPC over arithmetic circuits with malicious adversaries and an honest-majority. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS 2017). pp. 259–276 (2017), ePrint 2017/816
35. Malten, W., Ugurbil, M., de Vega, M.: More efficient comparison protocols for MPC (2023), preprint, ePrint 2023/1934
36. Möser, M., Eyal, I., Sirer, E.G.: Bitcoin covenants. In: Financial Cryptography and Data Security (FC 2016) Workshops. LNCS, vol. 9604, pp. 126–141 (2016)
37. Nuida, K., Ohata, S.: Efficient two-online-round secure comparison in the plain model (2026), ePrint 2026/808
38. Nunchuk: Open-source bitcoin agents with bounded authority. Nunchuk engineering blog (2026), `https://nunchuk.io/blog/bitcoin-agents`, accessed 2026-09-02
39. Panche, I.: Guardian: Role-gated MPC wallets for AI agents (2026), preprint, Zenodo 18684027
40. Porechna, D.: Threshold authorization without threshold signatures: Signature-agnostic MPC custody (2026), arXiv 2607.08226
41. Shamir, A.: How to share a secret. Communications of the ACM **22**(11), 612–613 (1979)
42. Taylor, J., Gerhart, P., Thyagarajan, S.A.K.: How to make delegated payments on bitcoin: A question for the AI agentic future. In: Financial Cryptography and Data Security (FC 2026) (2026)
43. Xiang, Z., Das, S., Li, Z., Ma, Z., Spiegelman, A.: The latency price of threshold cryptosystem in blockchains. In: Financial Cryptography and Data Security (FC 2025) (2025), arXiv 2407.12172
44. Yu, J., Zhou, S., Yin, H., Seong, B.: PASS: A provenanced access subaccount system for blockchain wallets (2026), arXiv 2604.22602
45. Zhang, Q., Chen, X., Zhang, Y., Chen, X.: SC-DT: Scalable constant round secure comparison and its application to privacy decision tree evaluation (2026), ePrint 2026/1389
46. Zhang, Y., Chen, X., Zhang, Q., Chen, X., Dong, Y.: Helix: Scalable multi-party machine learning inference against malicious adversaries (2025), ePrint 2025/347