HashDome: Bitcoin-Anchored Authentication for Contested Networks

Working Paper · Draft v0.3 · Dipl.-Ing. (RWTH) Georgios Nanos, MBA
Version 0.3 · May 25, 2026 · HashDome · Hamburg, Germany

Abstract. A decentralized form of network authentication would allow a node to be admitted to a network directly by its peers, without a continuously reachable central authority to issue or validate credentials at the moment of admission. Public-key signatures provide part of the solution, but the main benefits are lost if a trusted online party is still required to certify which keys are authorized whenever two nodes meet. We propose a construction that combines three costs, each anchored to an external proof-of-work chain: a credential delegated by a sponsor; that sponsor's stake, posted as a time-locked output on the external chain and attested in a committed trust state; and a proof-of-work receipt bound to a recent block of that chain. Forging an identity or flooding the network thereby imposes a cost measurable in committed capital and in energy. The trust state and the record of admissions are committed to the external chain through transactions whose inclusion is provable against block headers, forming an audit trail that cannot be altered without redoing the chain's proof-of-work. The system continues to authenticate while disconnected, relative to its last cached trust state, requiring only block headers and the committed-state proofs. We are explicit throughout about what the external chain can and cannot attest, and we collect the construction's limitations and open problems in Section 12.

Status of this document. This is an early-stage protocol sketch and concept note, not a finished cryptographic specification. Several mechanisms are described at the level of design intent and require formal treatment, parameter analysis, and independent review before implementation. Section 12 states the known limitations directly.

1. Introduction

Authentication on military and critical-infrastructure networks has come to rely almost exclusively on central authorities serving as trusted third parties to issue and validate credentials. While the model works well enough in garrison conditions, it inherits the weaknesses of the trust-based model. A certificate authority, a key-distribution center, or an identity provider is a single entity whose continued availability the entire network depends upon at the moment of admission. Authentication cannot proceed when that entity is unreachable, and an adversary who degrades or destroys it disables admission for every node at once. The cost of this dependence is paid precisely when it is least affordable: in the opening phase of a conflict, when the central authority is the first and most valuable target.

The dependence also spreads. To admit a new participant—a coalition partner, a newly deployed platform—its key material must be enrolled in advance, a procedure that presumes connectivity, prior coordination, and time. A certain fraction of operational delay is accepted as unavoidable, because the authority must be consulted to establish whether any given credential is genuine.

What is needed is an authentication system in which two authorized nodes can verify each other directly, without a live connection to any central service at admission time, and in which the evidence each presents is costly to forge and cheap to check. Admission that is computationally and economically impractical to forge would protect a network from infiltration, and a record of admissions, once committed, would be impractical to alter after the fact. In this paper we propose such a construction. Authorization is delegated through sponsors who commit verifiable stake; that stake and the resulting trust state are committed to an external timestamp chain—concretely, the

most computationally secured such chain in existence—through transactions whose inclusion any node can verify against block headers; and admission is gated by a proof-of-work receipt bound to a recent block of that chain.

We are deliberately careful about the word "trustless." This construction does not eliminate trust; it relocates it. Trust moves from a continuously reachable online authority to a set of pre-distributed root keys, the economics of sponsor stake, the immutability of the external chain, and locally cached state. The precise claim we defend is the following: *no continuously reachable online authority is required at the moment of admission*, and the trust that remains is made explicit, externally anchored, and economically costly to subvert. Section 12 states plainly where trust resides and where the construction's guarantees stop.

1.1 Doctrine Context

Bitcoin's proof-of-work has been characterized as a physical power-projection mechanism in cyberspace [Lowery, 2023]: energy irreversibly consumed in SHA-256 computation cannot be fabricated, reversed, or forged, creating a thermodynamic barrier that replaces trust-based access control with physics-based access control. Empirical analysis through 2026 finds that sovereign actors are treating this property strategically—establishing national Bitcoin reserves, competing for hash rate, and subsidizing mining operations independently of price signals [Carvalho, 2026]. HashDome applies this property at the authentication layer: the cost of admission is denominated in irreversible physical work, not in administrative privilege that an adversary can revoke or forge.

2. Authentication

We define a network admission credential as a chain of signatures. The network is founded by a root authority, which we treat not as a continuously trusted server but as a one-time genesis: a key, or a quorum of keys, whose public component is distributed to every node before deployment. The root signs a set of sponsors. Each sponsor in turn signs the nodes it vouches for. A verifier can follow the signatures back to the root to confirm that a presenting node has been authorized.

Let a node hold a key pair (sk_N, pk_N) on the curve secp256k1, the curve of the underlying chain. A sponsor S, itself authorized by the root, issues a credential

$$C = \big(pk_N,\ \texttt{class},\ h_{\text{iss}},\ h_{\text{exp}},\ pk_S,\ \sigma_S\big), \tag{1}$$

$$\sigma_S = \text{Sign}_{sk_S}\big(pk_N \,\|\, \texttt{class} \,\|\, h_{\text{iss}} \,\|\, h_{\text{exp}}\big), \tag{2}$$

where `class` is the access tier granted, and h_{iss} and h_{exp} are block heights of the external chain that bound the credential's validity in time. A verifier accepts C if σ_S verifies under pk_S, if pk_S is a sponsor authorized by the root and present in the current trust state (Section 4), and if the current chain height h satisfies $h_{\text{iss}} \le h < h_{\text{exp}}$.

The problem is that a verifier cannot tell from the signature chain alone whether a sponsor is scarce, or whether an adversary has manufactured authorization wholesale by obtaining or impersonating a sponsor key. This is the analogue, in authentication, of the double-spending problem: a signature can be copied at no cost, so possession of a syntactically valid credential proves nothing about its scarcity. A common solution is to consult a trusted central authority at every admission. The problem with that solution is the one identified in Section 1: the network's availability comes to depend on the authority being reachable.

We need a way for a verifier to know that a credential is scarce—that it could not have been manufactured cheaply and at scale—without consulting a reachable party at the time of admission. We accomplish this with two independent and externally anchored costs: a committed stake bound to

each sponsor and attested in a committed trust state (Sections 3–4), and a proof-of-work bound to each admission (Sections 5–6). The first makes authorization expensive to grant fraudulently; the second makes admission expensive to attempt at scale. Both are denominated in quantities—capital and energy—that an adversary cannot fabricate, and both are verifiable by any node against state it has cached from the external chain.

3. Sponsorship

To make authorization costly, we require each sponsor to bind its identity to a quantity of value committed on the external chain. The mechanism is a time-locked output, used here in the manner of a fidelity bond: value is rendered temporarily unspendable in a way that any party can verify, so that holding sponsor standing carries a real and measurable opportunity cost.

A sponsor S controls a key pk_S associated with one or more outputs holding total value v_S, each encumbered by a time lock of duration τ_S. The sponsor's standing is a function of this committed value and its lock:

$$w_S = f(v_S, \tau_S), \tag{3}$$

monotonically increasing in both arguments. The weight determines the tiers a sponsor may grant and the number of nodes it may vouch for: a node's admitted tier is capped by its sponsor's weight, and authority flows downward and cannot be amplified. A sponsor of weight w_S may authorize nodes whose aggregate tier demand does not exceed w_S; beyond that bound, additional credentials are rejected regardless of signature validity.

Binding standing to committed value inverts the economics of the Sybil attack. In the one-key-one-vote model, authority is subverted by anyone able to generate many keys, since key generation is free. Sponsorship is instead one-unit-of-stake-one-unit-of-authority. An adversary seeking sponsor weight W must commit value proportional to W, drawn from the same scarce asset that secures the external chain, and must lock it where the commitment can be observed. This is the established idea of the fidelity bond: an objectively verifiable sacrifice of liquidity that cannot be faked and that any party can confirm.

A long-term holder of the underlying asset can post such a commitment at little opportunity cost, provided it did not intend to move the value during the lock. The parties best able to become sponsors cheaply are therefore those with a durable stake in the asset, who have the least interest in degrading a network whose security rests upon it. An adversary, by contrast, must acquire and immobilize scarce value solely to attack, and forfeits its use for the duration.

4. Proof of Holdings and the Committed Trust State

This section corrects a claim that must be stated carefully, because it is the point on which a naive design fails. A sponsor's stake is a real on-chain output, but *a Bitcoin-style block header does not commit to the set of unspent outputs*. The Merkle root in a header commits to the transactions included in that block, not to the global state of which outputs remain unspent. Consequently a verifier holding only headers *cannot* prove, from the headers alone, that a given output exists and is presently unspent with a given value. Any construction that assumes otherwise is unsound. We therefore separate two questions that a naive design conflates.

What headers can attest. A header commits to the transactions in its block. Given a transaction tx and a Merkle branch ρ_{tx} to the transaction root M_h carried in the header at height h, a verifier holding that header can confirm

$$\mathrm{Verify}\big(M_h,\ \rho_{tx},\ tx\big) = \texttt{true}, \tag{4}$$

that is, that tx was included in the chain at height h. This is a genuine capability of header-only, simplified-payment-style verification.

What headers cannot attest. That an output created by tx has not since been spent. Spentness is a property of the evolving unspent-output set, which no header commits to. Confirming that an output is currently unspent requires either a full node, or an explicit external commitment to the relevant state.

The construction. HashDome introduces such an explicit commitment. Validators do not act individually. The set of currently valid sponsors and their weights is hashed into a *trust-state root* G_h, which is published only when a threshold of t-of-n designated validators independently verify each sponsor bond directly against full chain state—that it exists, is unspent, carries value v_S, and is locked for τ_S—compute the same trust-state root, and co-sign the `OP_RETURN` transaction using a threshold Schnorr scheme (FROST on secp256k1). An adversary controlling fewer than t validators cannot publish a fraudulent G_h regardless of mining power. The trust-state computation additionally deduplicates by UTXO outpoint: no two sponsors may claim the same committed output, closing the double-weight attack vector. The committing transaction is itself a transaction in a block, so its inclusion is provable against the header at h by exactly the capability of equation (4).

A node then verifies a sponsor in two steps, each of which the headers genuinely support:

1. the trust-state root G_h was committed on-chain at height h, shown by a transaction Merkle branch to the cached header, $\mathrm{Verify}(M_h, \rho_G, tx_{G_h}) = \texttt{true}$, where tx_{G_h} carries G_h;
2. the sponsor's entry $e_S = (pk_S, v_S, \tau_S)$ is a member of G_h, shown by a membership path ρ_S, with $\mathrm{Verify}(G_h, \rho_S, e_S) = \texttt{true}$.

The combined proof of holdings carried by a sponsor is therefore

$$\Pi_S = \big(e_S,\ \rho_S,\ G_h,\ \rho_G,\ h\big), \tag{5}$$

and the verifier recomputes $w_S = f(v_S, \tau_S)$ from the proven entry. Key control is shown separately by a signature on a challenge bound to the credential being issued,

$$\pi_{\text{key}} = \mathrm{Sign}_{sk_S}(r), \qquad r = \mathrm{H}\big(pk_N \,\|\, h_{\text{iss}}\big), \tag{6}$$

so the holdings proof is neither transferable to another credential nor reusable at another time.

What remains offline-unverifiable. A node isolated since height h cannot detect that a bond committed in G_h has since been spent or moved, nor that a newer trust-state root $G_{h'}$ with $h' > h$ has superseded it. Offline verification is thus always relative to the last cached trust state. This is the same staleness limitation that governs revocation (Section 7), and we treat the two uniformly: the construction authenticates against cached state, and the freshness of that state is bounded by the time since last synchronization. We do not claim, and the reader should not infer, that current spentness is provable from headers; it is not. Section 12 returns to the consequences.

Where revealing v_S is undesirable, the membership-and-threshold statement of step 2 may be produced in zero knowledge, demonstrating $v_S \geq v_{\min}$ without disclosing v_S or identifying the bond; this changes neither the weight semantics nor the verifier interface, and we treat it as a substitution at this step.

5. Proof-of-Power Admission

Stake makes authorization costly to grant; it does not by itself make admission costly to attempt. A node holding a valid credential could still flood the network with admission requests, or an

adversary could replay intercepted ones, at negligible cost. To impose a per-admission cost we require that each admission carry a proof-of-work whose form is that of an external-chain block header, and whose timestamp is bound to that chain, but whose *difficulty is an independent network parameter*. We distinguish two notions that must not be conflated.

Bitcoin-format proof-of-work — the 80-byte header structure and double-hash verification of the external chain, reused so that a receipt is checkable by a single hash operation and is anchored to a real block (Section 6). This is what HashDome adopts.

Bitcoin-network-difficulty proof-of-work — work at the external chain's *full* block difficulty. Requiring this of every legitimate admission would be impractical: an embedded radio, a drone, or a naval terminal cannot produce work at block difficulty in operational time. HashDome therefore does *not* set the admission target to the external chain's block difficulty.

The work is performed over a structure of the same form as a block header,

$$\beta = \big(\texttt{ver} \,\|\, b_{\text{prev}} \,\|\, m \,\|\, t \,\|\, \texttt{bits}_{\text{adm}} \,\|\, \texttt{nonce}\big), \tag{7}$$

in which b_{prev} is the hash of a recent block of the external chain, $\texttt{bits}_{\text{adm}}$ encodes the *admission target* T_{adm} set by network policy, and the field m, in the position of the Merkle root, commits to the admission:

$$m = \mathrm{H}\big(pk_A \,\|\, pk_B \,\|\, s\big), \tag{8}$$

with pk_A, pk_B the admitting and admitted nodes and s a session nonce supplied by pk_B. The prover varies `nonce` until

$$\mathrm{H}\big(\mathrm{H}(\beta)\big) < T_{\text{adm}}. \tag{9}$$

The expected number of evaluations is $2^{256}/T_{\text{adm}}$, tunable through $\texttt{bits}_{\text{adm}}$ and verified by a single hash. The admission target is chosen per platform class to balance the latency a legitimate node can tolerate against the cost an attacker must pay; we make this trade-off explicit in Section 11 and list its calibration among the open problems in Section 12. The reference to b_{prev} supplies the timestamp anchor (Section 6) and requires only the block hash, which headers provide; it is independent of the choice of T_{adm}.

The admission receipt presented by node A to node B is

$$\mathcal{R} = \big(C_A,\ \Pi_{S_A},\ \beta,\ s\big), \tag{10}$$

the credential, the sponsor's proof of holdings, the solved header, and the session nonce. Node B admits A if and only if all of the following hold, each checkable in a small constant number of hash and signature operations:

1. b_{prev} is a block in B's cached header chain;
2. b_{prev} lies within the last N blocks of that chain (Section 6);
3. $\mathrm{H}(\mathrm{H}(\beta)) < T_{\text{adm}}$ for the network's admission target;
4. $m = \mathrm{H}(pk_A \| pk_B \| s)$ for the s that B issued;
5. the credential chain of C_A verifies to the root, and Π_{S_A} proves w_{S_A} sufficient for C_A's tier against B's cached trust state;
6. the credential is unexpired: $h_{\text{iss}} \le h < h_{\text{exp}}$.

If all hold, the session is established without consulting any reachable authority. The verifier confirms the work with a single hash while the prover expended it in full, so the cost asymmetry runs

against an attacker: each forged attempt costs the full work, each rejection costs the verifier almost nothing. The work is required only at session establishment, not on the traffic that follows; the proof-of-power is a gate, not a tax on throughput, which confines its latency to the moment of joining.

6. Timelock Anchoring

The reference to b_{prev} does more than seed the work. Because the header committing to the admission contains the hash of a specific block, the receipt cannot have been computed before that block existed. The external chain is a distributed timestamp server: each block includes the previous block's hash, forming a chain in which a block cannot be altered without redoing its proof-of-work and that of every block after it. By embedding b_{prev}, an admission inherits this timestamp and proves it occurred no earlier than the moment b_{prev} was mined.

We use this to bound freshness and forbid replay. Let h be the verifier's chain tip and h_{prev} the height of b_{prev}. Condition 2 of Section 5 requires

$$0 \leq h - h_{\text{prev}} \leq N, \tag{11}$$

so a receipt is valid only within a window of N blocks; with a ten-minute mean interval, $N = 6$ admits a receipt for roughly one hour. A receipt against an older block is rejected, and one cannot be precomputed against a future block whose hash is unknown. Within the window, the session nonce s binds the receipt to a single challenge, so replay against the same verifier is detected by reuse of s.

The same anchoring gives the admission record an evidentiary property absent from conventional logs, which are only as trustworthy as the host that writes them. To place an admission at a different point in time, an adversary would have to rewrite the external chain from b_{prev} onward, redoing its accumulated proof-of-work and outpacing its honest hashing majority—the task whose probability Section 11 shows to vanish exponentially in depth. If an adversary controls a fraction q of the chain's hashing power and the honest majority $p = 1 - q$, the probability of rewriting from z blocks deep falls as $(q/p)^z$ for $q < p$. Anchoring to the chain of greatest accumulated work maximizes the depth z attained in any fixed time, and so minimizes this probability. This immutability concerns the *timestamp and the committed trust state*, which are recorded on-chain; it is independent of the admission difficulty T_{adm}.

In the partially synchronous model—where message delays are bounded by an unknown upper bound Δ—the common prefix property requires a depth parameter $k \geq 2\lambda f + 2\Delta$ rather than $2\lambda f$ [Garay et al., 2024, Theorem 35]. A battlefield network with $\Delta = 10$ blocks of maximum message delay therefore requires a minimum freshness window of $N = 6 + 2\Delta$ blocks, not 6. Operators deploying in high-latency contested environments must configure N accordingly.

7. Slashing and Revocation

The commitment of stake deters fraudulent authorization only if a sponsor that authorizes fraud can be made to suffer for it. A signature, once issued, cannot be recalled by its issuer; what the system can do is publish a sponsor's removal in a manner every node will observe and act upon. We treat revocation as a committed event. When the root determines that a sponsor S has acted in bad faith, it commits a revocation—in practice, by publishing a trust-state root $G_{h'}$ that omits S—to the external chain. On observing it, a verifier invalidates the entire subtree descended from S:

$$\text{revoke}(S) \implies \forall\, C \text{ via } S : \ \text{Verify}(C) = \texttt{false}. \tag{12}$$

A single committed update thus ejects every node S ever admitted, bounding the blast radius of a compromised sponsor to its own subtree.

The system cannot confiscate the committed value; no party but the holder of sk_S can move the output. What it imposes is forfeiture of everything the stake purchased: the value remains locked under τ_S, earning nothing, while the standing it conferred is destroyed and the key is marked, on an immutable ledger, as revoked. Let a sponsor hold value v_S, derive benefit b per unit time from honest participation, and let defection yield a one-time gain g. Honest behavior dominates whenever

$$g < \int_0^{\tau_S} b\, e^{-\delta t}\, dt \;+\; v_S\big(1 - e^{-\delta \tau_S}\big), \tag{13}$$

with δ the discount rate. Because the right-hand side scales with v_S, the threshold a sponsor must be tempted past grows with the very stake that qualifies it to sponsor at scale: the sponsors capable of the most authorization are those with the most to lose.

The honest limitation. Revocation while disconnected is bounded by what a node has received. A verifier rejects any credential in a revoked subtree present in its cached trust state, but a sponsor revoked *after* a node's last synchronization is unknown to that node, and a compromised sponsor or node may remain accepted inside an isolated partition until the partition receives the updated trust state. Revocation records are propagated peer-to-peer on a best-effort basis so that this knowledge spreads ahead of routine synchronization, but the fundamental property remains: *a disconnected node cannot act on a revocation it has not received*. This is shared by every offline credential system, and we state it plainly rather than imply it away. Its consequences are discussed in Section 12.

8. Disconnected Operation

The defining environment is one in which no central service can be reached and the external chain itself may be unreachable for extended periods. Each node retains, from its last connectivity: the block headers from genesis to the last synced height h^*; the most recent committed trust-state root G_{h^*} together with the transaction-inclusion proof linking it to a header; the membership data for sponsors and credentials relevant to its network; and the admission target T_{adm} in force. Headers are 80 bytes each, so the full history is on the order of tens of megabytes and the periodic update on the order of kilobytes per day—transmissible over any intermittent channel, including out-of-band ones.

With this state cached, a node performs the whole of admission verification (Section 5), timelock checking (Section 6), holdings verification against G_{h^*} (Section 4), and revocation checking against G_{h^*} (Section 7) with no query to the chain or to a server. What degrades under isolation is bounded and of three kinds: a wholly new enrollment, or a newly committed trust-state root, cannot be learned until the next contact; a sponsor bond spent or moved since h^* cannot be detected (Section 4); and a revocation issued since h^* cannot be acted upon (Section 7). In each case the failure is toward *stale acceptance relative to a known reference*, not toward unconditional acceptance: the structural guarantees—that a receipt carries real work, is bound to a real and recent block, and that its sponsor was attested in the cached trust state—remain intact.

The admission cost likewise does not collapse offline. The admission target T_{adm} is a fixed network parameter carried in cache, not a quantity that must be fetched, so an attacker who severs connectivity before attacking still faces the full per-admission work. This contrasts with a system whose admission depends on contacting a certificate authority or a revocation responder: deprived of that contact, such a system either fails closed, denying all admission, or fails open, admitting without check. The construction here does neither, because the evidence it requires was made portable in

advance and the cost it imposes is carried in the receipt. What it gives up in exchange is freshness, bounded by the time since h^*, as Sections 4 and 7 make explicit.

9. Reclaiming Trust State

The records committed to the external chain accumulate without bound. As with the spent transactions of the original timestamp construction, older records may be discarded once their effect is absorbed into a compact summary. The events relevant to authorization form, at each height, a current trust state: the authorized sponsors with their weights, and the nodes admitted under unrevoked credentials. A revocation cancels an enrollment; an expiry cancels a credential; a rotation supersedes a key. Once an enrollment is cancelled, the pair of records no longer affects any present admission and need not be retained in full—only their net effect on the current state.

That net effect is exactly the committed trust-state root G_h of Section 4. A node retains G_h and the proof of its inclusion, and may discard the superseded interior records, in the way interior Merkle hashes are stubbed once a branch is pruned. A credential's continued validity is then shown by a membership path to G_h rather than by replaying the history that produced it. The storage required of a node is thereby bounded by the size of the current trust state, not by the length of the network's history. The full, ordered record remains on the external chain for any party wishing to reconstruct history; what is pruned is the local obligation to store it, not the global record.

10. Simplified Admission Verification

A node need not retain the full state of the external chain. It keeps only the block headers of the longest proof-of-work chain—obtained as in simplified-payment verification by querying chain nodes until satisfied it holds the longest chain, then extended by the header updates of Section 8. From the headers alone it checks the two facts every admission's work depends on: that b_{prev} is in the longest chain, and that $\mathrm{H}(\mathrm{H}(\beta)) < T_{\text{adm}}$.

For the facts that reference committed state rather than work—the proof of holdings and a credential's membership in the current trust state—the verifier checks the inclusion of the committing transaction against a header and then membership within the committed root:

$$\mathrm{Verify}\big(M_h, \rho_G, tx_{G_h}\big) \;\wedge\; \mathrm{Verify}\big(G_h, \rho_S, e_S\big) \;\wedge\; \big(\mathrm{H}(\mathrm{H}(\beta)) < T_{\text{adm}}\big), \tag{14}$$

holding only headers and the supplied branches. The reliability of header-only verification is that of the original construction: it holds as long as honest nodes control the external chain, and is more vulnerable if the chain is overpowered, the condition whose probability Section 11 bounds. A node demanding stronger assurance—one admitting partners to the most sensitive tier—may run a full node of the external chain for independent and immediate verification, including the spentness checks headers cannot provide; the header-only method is offered for the constrained platforms whose storage and connectivity will not bear the full chain, and for which the headers, at tens of megabytes, are the practical limit.

11. Calculations

We consider three attacks: manufacturing identities at scale, forging an admission without its work, and rewriting the external chain to falsify a timestamp or a committed trust state. We bound the first two by quantities the adversary cannot fabricate; the third reduces to the attack analyzed in the original construction.

Sybil cost

Let admission to tier k require sponsor weight at least ω_k, and let the weight function be, for value v locked over duration τ,

$$w = f(v, \tau) = v\big(1 - e^{-\delta\tau}\big), \tag{15}$$

increasing in both arguments. An adversary authorizing n fraudulent nodes at tier k must control weight $n\,\omega_k$, hence commit value

$$v_{\text{adv}} \geq \frac{n\,\omega_k}{1 - e^{-\delta\tau}}. \tag{16}$$

The cost is linear in the number of identities and denominated in the scarce asset of the external chain. The committed value is public and time-locked, so it cannot be reused across simultaneous identities: a verifier checking Π_S against the committed trust state counts each bond once. The adversary's expenditure is the full v_{adv}, immobilized for τ.

Forgery cost

To present a receipt without performing its work, an adversary must find `nonce` with $\mathrm{H}(\mathrm{H}(\beta)) < T_{\text{adm}}$ more cheaply than search. Under the assumption that H behaves as a random function, no such means exists: the expected number of evaluations is

$$\mathbb{E}[\text{hashes}] = \frac{2^{256}}{T_{\text{adm}}}, \tag{17}$$

and an adversary of hashing rate H_{adv} produces valid receipts at rate $H_{\text{adv}}\,T_{\text{adm}}/2^{256}$, so to sustain λ admissions per second it must command

$$H_{\text{adv}} \geq \frac{\lambda\, 2^{256}}{T_{\text{adm}}}. \tag{18}$$

Here the admission target is the design lever. Let $W_{\text{adm}} = 2^{256}/T_{\text{adm}}$ be the expected work per receipt, and let a legitimate node of the weakest supported class compute at rate $H_{\min}$, so its admission latency is $W_{\text{adm}}/H_{\min}$ seconds. Choosing a tolerable latency L fixes

$$W_{\text{adm}} = L\,H_{\min}, \tag{19}$$

and the rate an attacker needs to sustain λ forged admissions is then $H_{\text{adv}} \geq \lambda L H_{\min}$; the attacker must out-compute λL of the weakest legitimate nodes. The link to the external chain's scale is optional and explicit: one may set $W_{\text{adm}} = \phi\,(2^{256}/T_{\text{btc}})$ as a fraction $\phi \ll 1$ of the external chain's per-block work, making the admission cost a stated fraction of block difficulty rather than an unstated one. The earlier error to avoid is implying $\phi = 1$; the practical regime is $\phi \ll 1$, and the security claim must be read in terms of W_{adm}, not block difficulty.

Indicative T_{adm} calibration by platform class

Platform	SHA-256 rate	ϕ	W_{adm} (hashes)	Latency
Embedded sensor	1 MH/s	1×10^{-16}	10^7	$\sim$10 s
Shipborne terminal	50 MH/s	5×10^{-15}	5×10^8	$\sim$10 s
Coalition laptop	500 MH/s	5×10^{-14}	5×10^9	$\sim$10 s

Calibration targets $\sim$10 s admission latency per platform tier. T_{adm} must be updated if adversary compute budgets change materially.

Rewriting the chain

The timestamp of Section 6 and the trust-state commitment of Section 4 are only as immutable as the external chain. With p the honest and $q = 1 - p$ the adversarial share of hashing power, the probability that an adversary z blocks behind ever catches up is

$$q_z = \begin{cases} 1 & \text{if } p \leq q, \\ (q/p)^z & \text{if } p > q. \end{cases} \tag{20}$$

A recipient who waits for z confirmations faces an attacker whose progress is Poisson-distributed with expectation $\lambda = z(q/p)$; the probability of a later catch-up is

$$1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} \left(1 - (q/p)^{z-k}\right), \tag{21}$$

dropping off exponentially in z. The construction inherits, without addition or weakening, the immutability of the chain it anchors to, for the timestamp and committed-state records it writes there.

The Bitcoin Backbone Protocol formally proves this result as the Common Prefix property [Garay et al., 2024, Theorem 15] under a precisely stated honest-majority assumption with parameter $k \geq 2\lambda f$. Chain Quality [Theorem 16] additionally bounds the fraction of adversarially committed trust-state updates in any sufficiently long epoch to $t/(n-t)$, where t is the number of corrupted Bitcoin miners. Even an adversary who controls $q < 1/3$ of network hash rate cannot insert fraudulent G_h commits at a rate exceeding $t/(n-t)$ of all epoch updates—and membership-proof verification against an honest sponsor set will reject any such fraudulent root.

Adversarial validator

An adversary controlling a minority of the t-of-n validator set cannot publish a fraudulent G_h. With a threshold co-signature requirement, compromise of fewer than t validators produces no valid `OP_RETURN` commit, regardless of Bitcoin mining power held. The combined adversarial cost therefore has three independent components: committed stake (capital), admission proof-of-work (energy), and accumulated chain work (computation)—none fabricable, all verifiable against cached headers.

Combined security

An adversary must defeat all three at once: commit v_{adv} in scarce, locked, attested value; command hashing rate $H_{\text{adv}} \geq \lambda\, 2^{256}/T_{\text{adm}}$ to admit at rate λ; and outpace the external chain's honest majority to falsify the committed record, with success $(q/p)^z$ vanishing in depth. The costs are independent and denominated in independent resources. The system is secure against these attacks as long as honest sponsors control more committed value than any cooperating adversary, the admission target is set so that H_{adv} exceeds an attacker's reach at the required λ, and the external chain's honest participants control a majority of its hashing power. It does not defend against an adversary who compromises a sponsor's private key and acts within an isolated partition before the revocation reaches it; that case is the subject of Section 12.

12. Known Limitations and Open Problems

We state the construction's limitations directly, both because intellectual honesty requires it and because each marks a problem that further work must close before the design can be called complete.

Headers do not commit to the unspent-output set. The single most important correction over a naive design: current spentness and value of a sponsor bond are not provable from block headers (Section 4). HashDome relies on an explicit trust-state commitment G_h, written on-chain via `OP_RETURN` and proven included by a transaction Merkle branch, with full validation performed by connected validators. This shifts a measure of trust onto those validators and onto the freshness of the committed state. A fully trust-minimized variant would require either a chain that commits to its unspent-output set, a succinct proof of that state (for example a zero-knowledge proof over the external chain), or a federated commitment with explicit accountability. We regard the precise commitment mechanism, its validator trust model, and its update cadence as open.

Admission difficulty is a calibration, not a constant of nature. The admission target T_{adm} must be low enough for the weakest legitimate platform to authenticate in tolerable time and high enough to price out an attacker at the required rate (Section 11). These two pressures may not always be jointly satisfiable on the most constrained hardware against the best-resourced adversary. The mapping from platform classes to admission targets, and whether a single network can span platforms of widely differing capability under one target, require empirical study.

Trust is relocated, not eliminated. The construction depends on pre-distributed root keys, the honesty and availability of validators that maintain the committed trust state, the economic assumptions behind sponsor stake, and the integrity of the external chain. "No continuously reachable online authority at admission time" is the accurate claim; "no trust" is not.

Offline freshness is bounded. Both stake validity (Section 4) and revocation (Section 7) are evaluated against the last cached trust state. A compromised sponsor or a spent bond may remain accepted inside an isolated partition until the updated state arrives. Best-effort propagation mitigates but cannot eliminate this; it is intrinsic to disconnected operation. Quantifying the residual risk as a function of partition duration and the value of stake at risk is open.

Key compromise. The proof-of-power and stake mechanisms raise the cost of forging and of Sybil attacks, but neither defends against theft of a legitimate sponsor or node private key. Detection and response then reduce to revocation, with the freshness bound above. Hardware key custody and rotation policy are out of scope here and material in practice.

Privacy and traffic analysis. Committing enrollment and trust-state events to a public chain creates an observable record. Even with hashed identifiers, the timing and volume of commitments may leak operational information; the zero-knowledge variant of Section 4 addresses value disclosure but not commitment timing. A treatment of metadata leakage and its mitigations is needed.

Formal analysis. The security arguments here are stated, not proved. A formal model of the admission protocol, a reduction of its guarantees to stated assumptions on H, the signature scheme, and the external chain, and machine-checked verification of the handshake remain to be done.

Post-quantum cryptography. The credential and sponsor signature layers use ECDSA/Schnorr on secp256k1, which is vulnerable to Shor's algorithm on a sufficiently large quantum computer. The system should be designed to permit signature-layer substitution. A concrete migration path would retain hash-based signatures at the root layer (for example XMSS or SPHINCS+) and replace secp256k1 node signatures with CRYSTALS-Dilithium once a compatible Bitcoin-layer profile is standardized. The proof-of-work and Merkle-branch components are quantum-resistant by construction.

13. Conclusion

We have proposed a construction for network authentication that does not depend on a continuously reachable trusted party at the moment of admission. We began with credentials made

from digital signatures, which control identity but cannot, alone, prevent the cheap manufacture of authorization—the analogue of double-spending. To resist it we required two externally anchored costs: a sponsor's stake, locked in scarce value and attested in a trust state committed to a public chain; and a proof-of-work receipt bound to a recent block of that chain, whose difficulty is a tunable network parameter and whose timestamp pins the admission to a record impractical to alter. We were careful to use the external chain only for what it can attest—transaction inclusion, and through it the inclusion of an explicit state commitment—and not for what it cannot, namely the spentness of an output from headers alone.

The construction admits nodes peer-to-peer, with no reachable authority consulted at admission, and continues to do so while disconnected, relative to a cached trust state whose freshness bounds its guarantees. Trust is not abolished but relocated, made explicit, and priced. The honest accounting of where that trust resides, and of where the guarantees stop, is given in Section 12; we regard those open problems as the substance of the work that remains, and this document as a sketch of a direction rather than a finished design.

References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. `https://bitcoin.org/bitcoin.pdf`

[2] J. Garay, A. Kiayias, N. Leonardos, "The Bitcoin Backbone Protocol: Analysis and Applications," *Journal of the ACM*, vol. 71, no. 4, August 2024. doi:10.1145/3653445.

[3] J. P. Lowery, "Softwar: A Novel Theory on Power Projection and the National Strategic Significance of Bitcoin," MIT Master's Thesis, February 2023.

[4] M. Carvalho, "Beyond Money, Hedge, and Energy: Evaluating Bitcoin as Power Projection Technology," ITA, SSRN-6259218, 2026.

[5] C. Ge, X. Ma, Z. Liu, "A semi-autonomous distributed blockchain-based framework for UAVs system," *Journal of Systems Architecture*, vol. 107, August 2020.

[6] A. Back, "Hashcash—A Denial of Service Counter-Measure," 2002.

[7] C. Dwork, M. Naor, "Pricing via Processing or Combatting Junk Mail," in *Advances in Cryptology—CRYPTO '92*, LNCS 740, pp. 139–147, 1993.

[8] R. C. Merkle, "Protocols for Public Key Cryptosystems," in *Proc. 1980 Symposium on Security and Privacy*, IEEE, pp. 122–133, 1980.

[9] S. Haber, W. S. Stornetta, "How to Time-Stamp a Digital Document," *Journal of Cryptology*, vol. 3, no. 2, pp. 99–111, 1991.

[10] C. Belcher, "Design for Improving JoinMarket's Resistance to Sybil Attacks Using Fidelity Bonds," 2019; see also BIP 46, "Fidelity Bonds."

[11] J. R. Douceur, "The Sybil Attack," in *Peer-to-Peer Systems (IPTPS 2002)*, LNCS 2429, pp. 251–260, 2002.

[12] Bitcoin developer documentation, "Block Chain" and "Simplified Payment Verification," *developer.bitcoin.org*.

[13] M. Campanelli, M. Hall-Andersen, S. H. Kamp, "Curve Trees: Practical and Transparent

Zero-Knowledge Accumulators," USENIX Security, 2023; and related anonymous unspent-output ownership work, 2024.

[14] W. Feller, *An Introduction to Probability Theory and Its Applications*, Wiley, 1957.

[15] NIST, "FIPS 180-4: Secure Hash Standard (SHS)," 2015.