zkBTC: Fully Private, Trustless Bitcoin

TaprootFreak

www.zkcoins.com

Abstract. A purely peer-to-peer version of private electronic cash backed by bitcoin would allow payments to be sent without revealing amounts or the transaction graph and without a custodian. Digital signatures and zero-knowledge validity proofs provide part of the solution, but the main benefits are lost if a trusted party is still required to order transactions or to hold the backing bitcoin. We propose a system for private electronic cash that uses Bitcoin itself only as an ordering layer. zkCoins is a client-side-validated transfer protocol: each state transition carries a recursive validity proof, and the only on-chain footprint is a constant-size nullifier of 64 bytes that prevents double-spending without a global ledger of amounts. zkBTC is a one-to-one bitcoin-backed token on that protocol. Its reserve is held in vaults spendable only along pre-signed BitVM2 fraud-proof paths. Anyone may join the operator set; that set only grows; every new vault is signed by all current operators, so a holder who registers before the first mint (before that vault's setup) is the honest signer of every vault and can also serve their own exit. A gatekeeper, when designated, may screen new deposits and cannot freeze, seize, or block exit. Under the stated honesty and liveness assumptions (one-of-N setup honesty, at least one honest live challenger acting within each challenge window, and sound circuit and graph cryptography), no operator can steal the reserve. zkBTC is effectively trustless in that sense: you are an operator; you trust nobody else with your bitcoin.

1. Introduction

Internet commerce still settles through financial institutions acting as trusted third parties. Bitcoin removed that trust for settlement, yet every payment on its ledger broadcasts the payer, the payee, and the amount to the world. Confidentiality is routinely required for ordinary commerce; the accepted remedies give up what makes Bitcoin valuable. Separate privacy chains replace Bitcoin's security model with a new one. Custodial or federated wrapped tokens reintroduce exactly the trusted third party Bitcoin removed.

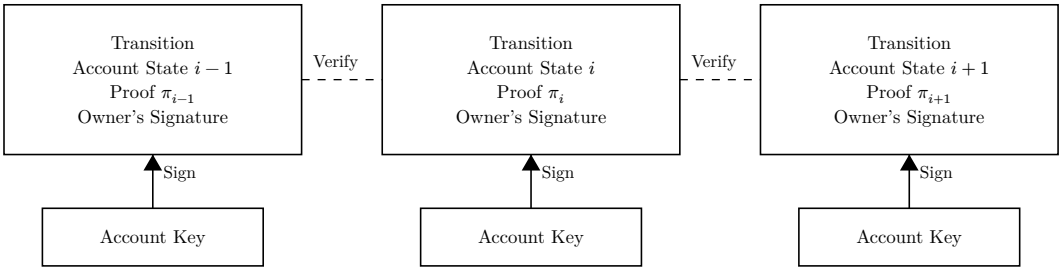
What is needed is an electronic cash system that keeps Bitcoin as the only settlement layer and the only backing asset, hides amounts and the transaction graph cryptographically, and lets any holder exit back to on-chain bitcoin without anyone's permission. This paper proposes such a system. zkCoins is a client-side-validated transfer protocol whose only on-chain footprint is a constant 64 bytes per transaction. zkBTC is a bitcoin-backed token on that protocol whose reserve is secured by fraud proofs rather than custody. Honest majority hashpower orders the nullifiers. Operators cannot steal the reserve as long as one honest operator per backing group deletes its epoch signing key at setup, at least one honest live challenger acts within each challenge window, and the circuit and BitVM2 graph cryptography are sound. The holder who joins the cumulative operator set before the first mint *is* that honest operator for every vault. Mint canonicity (a

private-fork mint) is a separate residual; a designated gatekeeper can check it at mint time only. A gatekeeper is not what makes zkBTC trustless.

2. Transactions

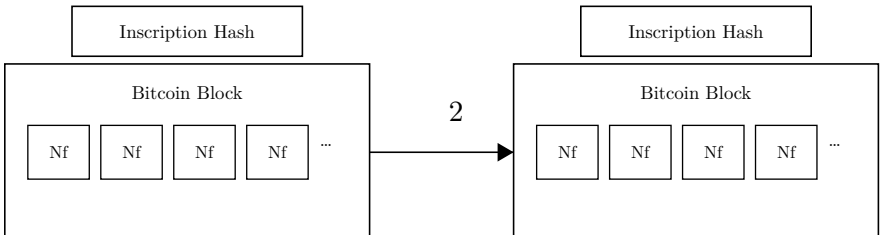
We define an electronic coin as a chain of proof-carrying account-state transitions. Each account is a sequence of states. Each transition (send, receive, mint, or redeem) carries a zero-knowledge proof that it follows the protocol rules given its predecessor, and a BIP-340 signature by the account's current signing key. The recipient verifies the proof. Verification is client-side: nobody else needs to see amounts or parties. The payee verifies the chain of proofs the way a Bitcoin payee verifies the chain of signatures.

The problem is that the payee cannot tell that the payer did not create a second, conflicting transition of the same account state. A central mint could order transitions, but then the whole money system depends on that mint, and the privacy and exit properties collapse back into custody. We need the payee to know that no earlier conflicting transition exists. For that there must be a single agreed history of transition markers, publicly announced, without revealing the amounts or parties those markers stand for. For zkBTC the first transition of a coin is a mint bound to an on-chain vault deposit (Section 11) and the last is a redeem that burns it; between the two, transfers are ordinary zkCoins transitions, and the backing never moves. The lifecycle is therefore mint, transfer, redeem: only the endpoints touch the reserve, and every internal step stays off the Bitcoin ledger except for its nullifier marker.



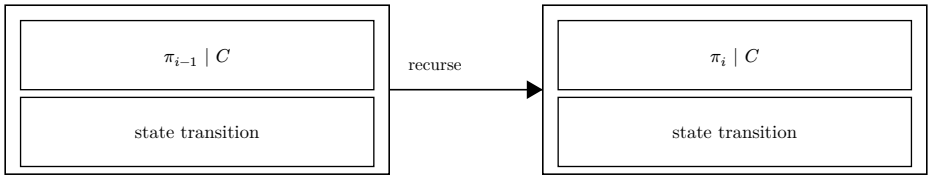
3. The Nullifier Chain

The solution begins with Bitcoin itself as the ordering layer. Bitcoin is the timestamp server. Every state-advancing transition publishes a constant-size nullifier (a pair (Pk_i, R_i), an account-key-derived public key and a signature nonce commitment, 64 bytes) into Bitcoin blocks via inscriptions, half-aggregated per batch. The nullifier reveals nothing about amounts, parties, or the transaction graph; it is meaningful only to those who already hold the coin's history. The first occurrence of a given key Pk_i on the chain is the one that counts. A second conflicting transition of the same state hits the same key and is rejected by every verifier. In-circuit predecessor-anchoring makes each transition prove that its predecessor's nullifier is already on-chain, so no off-chain fork can survive. Half-aggregation lets a publisher combine many transitions' nullifier signatures into a single inscription that carries one shared aggregate scalar, so the on-chain cost per transition stays at the 64-byte pair even when a batch holds many markers.



4. Validity Proofs

Where Bitcoin enforces rules with proof-of-work, zkCoins enforces them with zero-knowledge validity proofs. One compliance predicate C (a single circuit) checks every transition: per-asset value conservation (wide-sum, overflow-safe, so no transition creates value), authorization by the account key, correct predecessor state, one-time use of each coin, and the anchoring of everything the verifier cannot see into hiding commitments. Proofs compose recursively (proof-carrying data): verifying the latest proof verifies the entire history back to genesis, so verification cost is constant regardless of history length. Forging a coin means breaking the proof system, not out-computing the network. The circuit is fixed and its digest pinned: everyone verifies against the same predicate, like every Bitcoin node checks the same proof-of-work target.



5. Network

The steps to run the network are as follows:

1) The sender builds a state transition and its validity proof, and sends the coin proof directly to the recipient off-chain (gift-wrapped, NIP-17 style transport).
2) The sender submits its nullifier to a publisher.
3) Publishers batch nullifiers and inscribe the batch into a Bitcoin block, paying on-chain fees.
4) Bitcoin orders the inscriptions with proof-of-work.
5) Recipients (their nodes) scan blocks and accept a transition only if its nullifier's first occurrence matches.
6) Wallets extend their account chains on top of anchored transitions.

Nodes always consider the Bitcoin chain with the most cumulative proof-of-work to be the correct one. Reorganizations follow Bitcoin's own longest-chain rule: a transition buried deeper than the finality depth is settled. If two inscriptions of the same key appear, only the first occurrence in the canonical chain is admitted; the later is a double-spend and is rejected.

Publishers are permissionless and interchangeable. Anyone can self-publish. New nullifier announcements are tolerant of delayed broadcast: a recipient who misses a batch can still verify later against the chain. A nullifier does not need to reach every publisher; one inclusion in any batch that confirms is enough, and a sender who distrusts all publishers inscribes it itself. No

publisher can steal: they never touch coins, only 64-byte markers. Censoring a specific user costs fees lost to a competitor. Forging is out of reach: the markers carry no spendable value, and validity is enforced by the proofs the recipient checks. Ties between Bitcoin forks are resolved by waiting for further proof-of-work, as in Bitcoin itself; nullifier first-occurrence is always evaluated on the surviving canonical chain after reorg.

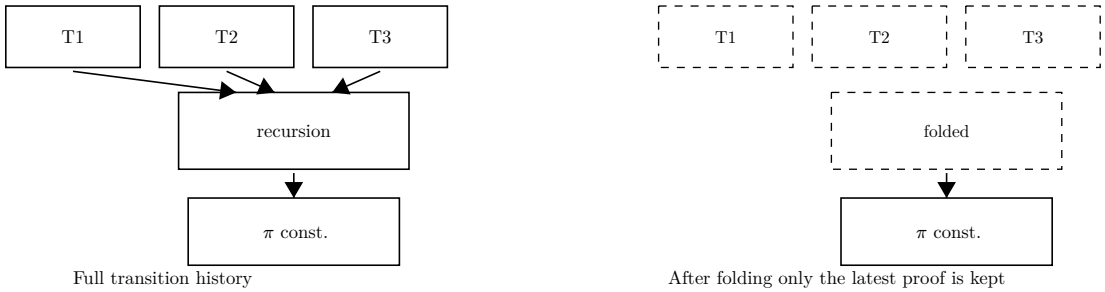
6. Incentive

The protocol defines a fee-coin mechanism under which publishers are compensated per inscribed marker. They front the on-chain inscription cost and are paid in fee coins attached to batch entries. The first protocol version keeps publishing sponsored: anyone can self-publish or use a sponsoring publisher. The open fee market is a defined, deferred mechanism rather than a launch feature; there is no new issuance for publishers. The arrangement is analogous to miners collecting transaction fees once block subsidy declines.

For the zkBTC reserve, operators post bonds and earn redemption fees (the holder's max_fee ceiling) for fronting bitcoin to redeemers. A false assertion is disproved on-chain, the operator's bond is slashed and the claim is void, so a cheater burns its stake for a claim that pays nothing. Challenging is permissionless, and how challengers are funded is an economic calibration the design leaves open. Honesty is the profitable strategy. A publisher gains nothing by withholding: users switch. Under those rules and the stated assumptions (one-of-N setup honesty, at least one honest live challenger in-window, and sound circuit and graph cryptography), cheating is not profitable. The incentive is fee income for honest service, not issuance of a new base asset. An attacker who accumulates bonds and registers many operators gains no direct spend power over the vaults; spends exist only along the pre-signed graph. Profit-seeking capital is better spent serving redemptions for fees than burning bonds on disprovable assertions. The greedy strategy is honest service, not coordinated fraud.

7. Compact State

Once the latest recursive proof of an account is verified, the earlier proofs need not be kept. Bitcoin prunes spent transactions after the fact; zkCoins never materializes the global history at all. Recursion folds the entire transition history into one constant-size proof. The chain carries only 64 bytes per transaction. Nodes keep no global UTXO set and no per-user state: an append-only log of first-occurrence markers suffices, and every account's history folds into a constant-size recursive proof.



Full transition history

After folding only the latest proof is kept

A rough capacity bound follows from block space alone. With 4 MB blocks every 10 minutes and 64 bytes per nullifier, Bitcoin as-is admits on the order of $4 \times 10^6/64/600 \approx 104$ transitions per

second if the entire block were nullifiers. Block verification for the nullifier stream is cheaper than Bitcoin's own: there is no witness execution and no UTXO lookup. New nodes need no initial block download of transaction data, only the 64-byte markers and the append-only first-occurrence log. Every node derives that log from the chain alone, so nodes can leave and rejoin at will, accepting the nullifiers inscribed while they were gone as proof of what happened — there is nothing else to trust.

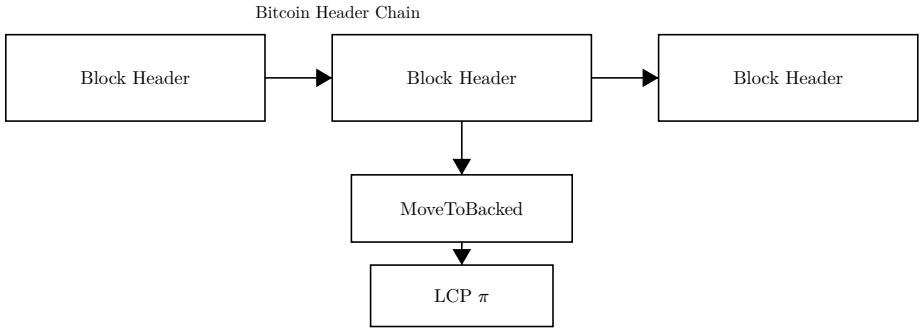
8. Simplified Verification

It is possible to verify payments without running a full proving stack. Two light-verification arrangements apply.

A thin wallet may hold only its keys and secret blinds while a node runs scanning and proving. The account's next spending key is bound through a wallet-native hiding commitment, so a malicious node cannot rotate the wallet's keys. It can, however, still choose the outputs it proves: binding payment intent into the circuit is documented open design work, so a wallet that delegates proving places that much trust in its node.

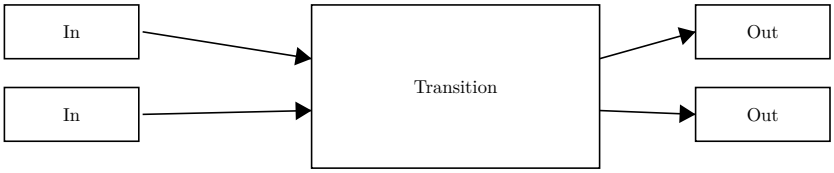
Separately, the mint transition embeds a recursive Bitcoin light-client proof (headers and proof-of-work depth) showing that the backing transaction is buried at least D_{mint} blocks deep. This is in-circuit simplified payment verification. As Nakamoto noted for SPV, proof-of-work depth proves work, not canonicity: a private fork can also be deep. That residual is exactly why the mint path can carry an external canonical-view check (the gatekeeper of Section 11).

Users who receive frequent or high-value payments will still prefer to run their own node and prover for independent verification. Delegation is a convenience with that residual; running both recovers the full client-side verification path of Section 4 without relying on a third party for scanning or proving.



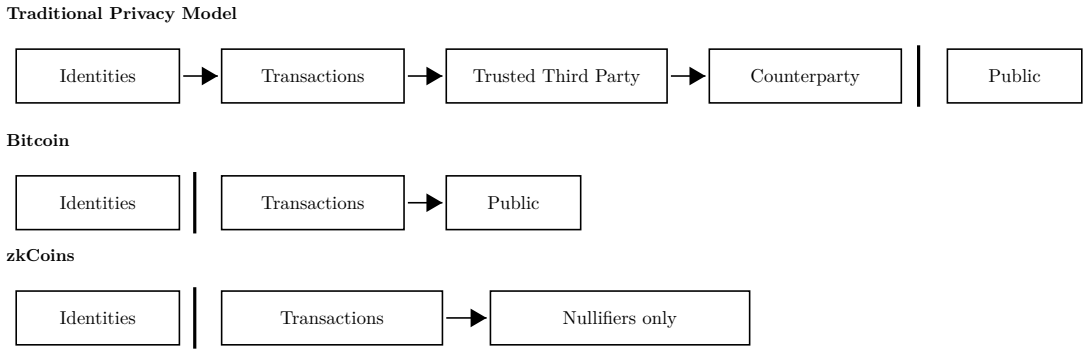
9. Combining and Splitting Value

Although it would be possible to handle coins individually, transitions that contain multiple inputs and outputs are more efficient. A transition admits up to eight inputs and eight outputs; conservation is enforced in-circuit. A typical payment takes one input coin and splits it into a payment output and a change output. There is never a need to extract a standalone copy of a coin's full history; recursion has already folded it into the latest proof.



10. Privacy

Banks provide confidentiality by gatekeeping information: access is limited to the parties and their intermediary. A system that must announce transition markers publicly has to create privacy elsewhere. Bitcoin breaks the information flow at identities, keeping them outside the public ledger while amounts and the graph remain fully public. zkCoins moves the boundary further: only the 64-byte nullifiers are public. Amounts, parties, and the graph stay between sender and receiver, protected by hiding commitments with fresh blinds. Address reuse does not link on-chain markers; each transition's nullifier key is one-time.



As an additional firewall, a new account key is used for each transition, and common practice is a new receiving account per counterparty relationship when unlinkability among counterparties is required. The honest boundary remains: peg-in and peg-out of zkBTC are ordinary public Bitcoin transactions, so amounts and timing at the boundary are observable and correlatable. Internal transfers are shielded.

11. The Bitcoin Reserve

We now describe the bitcoin-backed token. zkBTC is token standard 3 on the zkCoins transfer system: a one-to-one claim on bitcoin held in vaults, not in custody.

The vault is a taproot output with a NUMS internal key, so there is no key-path spend. Its only spend paths are an ordered, N-of-N pre-signed BitVM2 transaction graph (assert, challenge, disprove, payout) fixed at deposit setup. After setup the epoch signing keys are deleted; identity keys are not. The vault then has no live signer for that epoch's graph. Under one-of-N honest epoch-key deletion, no coalition can sign anything outside the graph.

Peg-in (mint) proceeds as follows. The depositor and the operators co-sign a MoveToBacked transaction that places the deposit under the vault. The mint transition proves in-circuit, via the light-client proof of Section 8, that MoveToBacked is buried at least D_{mint} blocks deep (deep finality, on the order of 2016 blocks) and that the vault instance matches the asset's terms: the operator-registration policy root, amount equality with the vault output, and a one-shot mint key so each vault mints exactly once. Circulating supply is then a conditional upper bound against the public vault set on the canonical chain, not an unconditional invariant of the chain alone.

The operator set is open, permissionless, and growth-only. Anyone may join by posting a bond with proof of key possession. Nobody leaves. Every new vault is N-of-N over the current full set.

A holder who registers before the first mint is the honest signer of every vault: a later Sybil club cannot omit them, and they will not co-sign a graph that pays a vault to a thief. That closes the self-controlled-operator drain (Attack B). Join before the first mint; coins minted earlier are an untrusted prefix because the token is fungible.

The gatekeeper is optional and per-asset. Pure in-circuit proof-of-work depth cannot prove canonicity (a private fork can be deep). That is Attack A, a Bitcoin-class residual mitigated by deep finality on the order of 2016 blocks, closed cleanly only if a designated gatekeeper withholds its mint signature until it has confirmed, on its own canonical Bitcoin view, the backing transaction and the registration commitment. It has no key on the vault, no role in transfers or redemption, and cannot freeze, seize, or redirect. It can only refuse new mints. It is not the Attack-B close and not what makes zkBTC trustless. A negligent gatekeeper that skips the canonical check enables Attack A, not a Sybil drain of later vaults.

Peg-out (redeem) is burn-first. The holder's redeem transition destroys the coin and publishes a redeem identifier (its transition nullifier) committing the payout address and a max_fee ceiling in hiding form. There is no un-burn: a ceiling set too low can leave a burned coin unserved. Any registered operator fronts the payout from its own funds within max_fee, then reclaims exactly the redeem amount from the vault by asserting correct service through the BitVM2 graph. Any watcher can challenge a false assertion within the challenge window; a single successful disprove slashes the operator's bond and voids the claim. Reimbursement is claimant-committed, sole-funded, and single-party-authorised: the payout transaction is non-transferable by construction. Under the stated assumptions (one-of-N setup honesty, at least one honest live challenger acting within the window, and sound circuit and graph cryptography), the worst case is a freeze (redemption waits for a live operator), not theft. A critical circuit or graph soundness bug is the theft case. Sound circuit and graph cryptography is a standing assumption.

Honest residuals remain even when theft is closed. The depositor's co-signature makes a vaulted contribution consented; a gatekeeper that withholds its mint signature after MoveToBacked has fired leaves that deposit an irrevocable reserve contribution, a depositor-side loss path. With no council and no admin keys, a freeze under the stated assumptions can be permanent absent a future covenant upgrade. "Not theft" does not mean "recoverable."

We consider the probability of a canonical-chain catch-up in the same terms as Nakamoto's attacker analysis [1]. The race between the honest chain and an attacker is a Binomial Random Walk. The probability of an attacker catching up from z blocks behind, when p is the probability an honest node finds the next block and $q = 1 - p$ is the attacker's, is the Gambler's Ruin probability

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

The verifier of a new mint cannot know the attacker's progress on a private fork. How long must the network wait before treating a mint's backing as settled? Assuming honest blocks took the expected time, the attacker's potential progress while the mint waits for z confirmations is a Poisson distribution with expected value

$$\lambda = z \cdot q/p.$$

To get the probability the attacker could still catch up now, multiply the Poisson density for each amount of progress by the catch-up probability from that point:

$$\sum_{k=0}^{\infty} \left(\lambda^k e^{-\lambda}/k!\right) \cdot \begin{cases} (q/p)^{(z-k)} & \text{if } k \le z \\ 1 & \text{if } k > z. \end{cases}$$

Rearranging to avoid summing the infinite tail of the distribution:

$$1 - \sum_{k=0}^{z} \left(\lambda^k e^{-\lambda}/k!\right) \cdot \left(1 - (q/p)^{(z-k)}\right).$$

Converting to C code:

```
double AttackerSuccessProbability(double q, int z)
{
  double p = 1.0 - q;
  double lambda = z * (q / p);
  double sum = 1.0;
  int i, k;
  for (k = 0; k <= z; k++)
  {
    double poisson = exp(-lambda);
    for (i = 1; i <= k; i++)
      poisson *= lambda / i;
    sum -= poisson * (1 - pow(q / p, z - k));
  }
  return sum;
}
```

Running some results, the probability drops off exponentially with z:

```
q=0.1
z=0 P=1.0000000
z=1 P=0.2045873
z=2 P=0.0509779
z=3 P=0.0131722
z=4 P=0.0034552
z=5 P=0.0009137
z=6 P=0.0002428
z=7 P=0.0000647
z=8 P=0.0000173
z=9 P=0.0000046
z=10 P=0.0000012

q=0.3
z=0 P=1.0000000
z=5 P=0.1773523
z=10 P=0.0416605
z=15 P=0.0101008
z=20 P=0.0024804
z=25 P=0.0006132
z=30 P=0.0001522
z=35 P=0.0000379
z=40 P=0.0000095
z=45 P=0.0000024
z=50 P=0.0000006
```

Solving for P less than 0.1%:

```
P < 0.001
q=0.10 z=5
q=0.15 z=8
q=0.20 z=11
q=0.25 z=15
q=0.30 z=24
q=0.35 z=41
q=0.40 z=89
q=0.45 z=340
```

Even the extreme row ($q = 0.45$) needs only $z = 340$ for $P < 0.1\%$. Mint settlement waits $z = D_{\text{mint}} \approx 2016$, where the pure catch-up probability collapses further. The closed form $q_z = (q/p)^z$ for $p > q$ at deep finality yields, for three attacker shares:

```
q=0.10 (q/p = 1/9): z=2016 P ≈ 10^-1924
q=0.30 (q/p = 3/7): z=2016 P ≈ 10^-742
q=0.45 (q/p = 9/11): z=2016 P ≈ 10^-176
```

At deep finality catch-up against the honest chain is economically closed even for a 45% attacker: $z = 2016$ yields probabilities on the order of 10^{-176} and smaller. That is not Attack A: a private fork can still be deep without catching the honest chain. The residual mint risk is therefore canonicity of the proven chain, not raw proof-of-work catch-up. A designated gatekeeper can check canonicity at mint time. The Sybil-operator drain is closed by the growth-only operator set, not by the gatekeeper.

12. Conclusion

We have proposed a system for private electronic cash backed by bitcoin without custodial trust. Coins are chains of proof-carrying state transitions. Bitcoin orders constant-size nullifiers so double-spends are publicly detectable without revealing amounts. Validity proofs replace global validation of a ledger of values; the on-chain footprint stays constant. The reserve is secured by pre-signed fraud-proof paths with an open, growth-only operator set: a holder who registers before the first mint is the honest signer of every vault and can serve their own exit. Under one-of-N setup honesty, at least one honest live challenger in each window, and sound cryptography, operators cannot steal. That is the sense in which zkBTC is effectively trustless. A gatekeeper, when designated, acts only at the mint boundary and cannot freeze transfers or redemption. Exit depends on liveness of some registered operator, never on permission. The guarantees are conditional on the enumerated assumptions. This paper is an informative introduction; the design is specified normatively in [8] and awaits implementation.

References

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

[2] R. Linus, "zkCoins," https://gist.github.com/RobinLinus/d036511015caea5a28514259a1bab119, 2023.

[3] J. Nick, L. Eagen, R. Linus, "Shielded CSV: Private and Efficient Client-Side Validation," Cryptology ePrint Archive 2025/068, 2025.

[4] P. Todd, "Scalable Semi-Trustless Asset Transfer via Single-Use-Seals and Proof-of-Publication," https://petertodd.org/2017/scalable-single-use-seal-asset-transfer, 2017.

[5] R. Linus et al., "BitVM2: Bridging Bitcoin to Second Layers," https://bitvm.org/bitvm_bridge.pdf, 2024.

[6] E. Bal, L. Aumayr, A. İyidoğan, G. Scaffino, H. Karakuş, C. E. Aslan, O. S. Thyfronitis Litos, "Clementine: A Collateral-Efficient, Trust-Minimized, and Scalable Bitcoin Bridge," Cryptology ePrint Archive 2025/776, 2025.

[7] P. Wuille, J. Nick, T. Ruffing, "Schnorr Signatures for secp256k1," BIP-340, 2020.

[8] "zkCoins protocol specification," https://github.com/zk-coins/docs, and "zkBTC token standard," https://github.com/zk-BTC/zkbtc, 2026.