Multisig Bearer Instruments: Non-Custodial Bitcoin Transfer

Asensio Arias
keys@multisigbearer.org multisigbearer.org
DOI: 10.5281/zenodo.20066812

Abstract

A non-custodial threshold instrument for Bitcoin would allow value to transfer between parties without network connectivity, fees, or custodial dependency. Digital signatures and multisignature scripts provide part of the solution, but the core benefit is lost if the issuer retains a key capable of unilateral redemption. All prior multisignature schemes have positioned the issuer at or above the spending threshold. We propose a system that inverts this: the holder receives the two keys constituting the spending threshold of a 2-of-3 multisignature script, and the issuer holds one key arithmetically below it. The issuer's key is deleted before the instrument is funded. No copy exists during the instrument's funded lifetime. Transfer is key handover requiring no on-chain transaction, no network connection, and no fee. Only two on-chain transactions occur across the instrument's lifetime regardless of transfer count. We describe a buyer-generated key issuance protocol that closes the malicious issuer attack, and a receipt mechanism using the Nostr protocol that provides cryptographic proof the receiver holds the spending keys, triggering deletion of the sender's copies. We characterise the trust model of software-only deployments honestly, distinguishing cryptographic guarantees from economic ones, and describe the TEE path that converts economic to cryptographic assurance.

---

1 Introduction

Commerce on Bitcoin requires an on-chain settlement event for every transfer of value. Fees, confirmation latency, and network dependency follow necessarily from this. While the base layer provides strong finality guarantees, these properties make it unsuitable for small, frequent, or physically proximate payments. The Lightning Network reduces fees and latency through payment channels but requires channel liquidity, online counterparties, and network connectivity at payment time. A gap remains for payments, like buying a coffee, that are small, frequent, or made without network access, where the costs and dependencies of both mechanisms become impractical.

Physical key-based instruments for Bitcoin have been attempted in several forms since the early years of the protocol. In each case, the issuer held a private key at some point during manufacture or issuance. Single-key designs create an unresolvable dependency on the issuer's non-retention. Multisignature threshold schemes have addressed part of this by distributing keys between parties, but in every prior implementation the issuer has been positioned at or above the spending threshold, either as a sole keyholder or as a required co-signer. The holder's claim on the funds has always been contingent on the issuer's behaviour or continued availability.

The present proposal inverts this: the holder receives both keys required to meet the spending threshold, and the issuer holds one sub-threshold key that is arithmetically incapable of spending alone. The issuer's key is deleted immediately after address construction and before any Bitcoin is sent to the address, making the lockout permanent and unconditional. This issuance property is verifiable by anyone who can derive the P2TR address from the three public keys and confirm it matches the funded output.

The harder problem is transfer. Once the holder decides to pass the instrument on, they must transmit their two spending keys to the receiver, then delete their own copies. The receiver needs confidence the keys work; the sender needs confidence the receiver holds them before deletion. Neither party can fully verify the other's software behaviour. This paper describes a transfer protocol that provides cryptographic proof the receiver holds the keys, and characterises honestly what that proof does and does not guarantee about the sender's deletion.

Once funded, the instrument transfers through key handover. No transaction is broadcast, no fee is paid, no network connection is required. The UTXO remains at the same address through any number of transfers. The final holder sweeps to their own address at their discretion. That sweep is the second and last on-chain event in the instrument's lifetime.

2 Background and Prior Work

2.1 Physical Key-Based Bitcoin Instruments

Several approaches to physical Bitcoin key-transfer instruments have been developed since the early years of the protocol. Single-key designs encode a private key in a physical medium and rely on the physical form as a signal that the key has not been accessed since manufacture. The issuer necessarily generates and handles the private key before delivery. Hardware-enforced write-once designs improve on this by generating the key on-device and revealing it only on physical destruction, but remain limited by single-key architecture. In each case the security claim rests on a statement about the issuer's behaviour rather than on a verifiable property of the key distribution.

2.2 Multisignature Bitcoin

Bitcoin's scripting system supports m-of-n multisignature addresses, which require m valid signatures from a designated set of n keys before funds can be spent. Pay-to-Script-Hash encapsulated these conditions such that the redeem script was revealed only at spend time. Pay-to-Taproot replaces this with a Schnorr-based key-path spend and an optional Tapscript tree, eliminating the on-chain multisig fingerprint. Multisignature has been applied to collaborative custody and exchange cold storage. Prior multisignature threshold schemes have positioned the issuer as a required co-signer, preserving custodial dependency through the instrument's settled lifetime. The present proposal keeps the issuer permanently below the spending threshold and deletes the issuer's key before funding. The issuer is not a co-signer. The issuer is not required for settlement. The issuer cannot act on the instrument after creation.

2.3 Off-Chain Payment Schemes

Chaumian ecash systems issue bearer tokens using blind signatures and allow off-chain transfer without per-transaction settlement costs. Both issuance and redemption require the mint as a trusted and online participant. Lightning Network channels allow off-chain payments between nodes with established channels but require liquidity management, channel funding transactions, and online counterparties at payment time. The mechanism proposed here is complementary to Lightning: instruments can be swept into Lightning channels at settlement, and this protocol serves payment contexts where Lightning cannot operate.

3 The MBI Mechanism

3.1 Core Architecture

We define a Multisig Threshold Instrument as a funded unspent transaction output locked to a Pay-to-Taproot (P2TR) address where the holder controls exactly m keys and the issuer holds $(n - m)$ keys, with $m > (n - m)$. In the recommended configuration, $n = 3$ and $m = 2$.

Key A is held by the issuer. Keys B and C are held by the instrument holder. The internal key of the P2TR address is the BIP-327 MuSig2 aggregate of K_B and K_C. The holder, controlling both B and C, can spend via the key path using a single aggregate Schnorr signature. The issuer, holding only A, cannot produce a key-path signature because K_A is not part of the aggregate key. Key A appears only in Tapscript leaves within the script tree, where a 2-of-3 `CHECKSIGADD` leaf and a `CLTV` recovery leaf provide fallback spending paths.

This threshold configuration is the central departure from all prior multisignature schemes used in this context. In prior art, the spending function includes at least one issuer key as a required input. In the present scheme, spend $= f(k_B, k_C)$. The issuer's key participates only in address construction, after which it is deleted immediately and before any Bitcoin is sent to the address. A key-path spend using Keys B and C alone produces a witness indistinguishable from any single-key P2TR transaction. The script tree exists only as a commitment in the tweak; it is never revealed during normal operation. This is arithmetic, not policy.

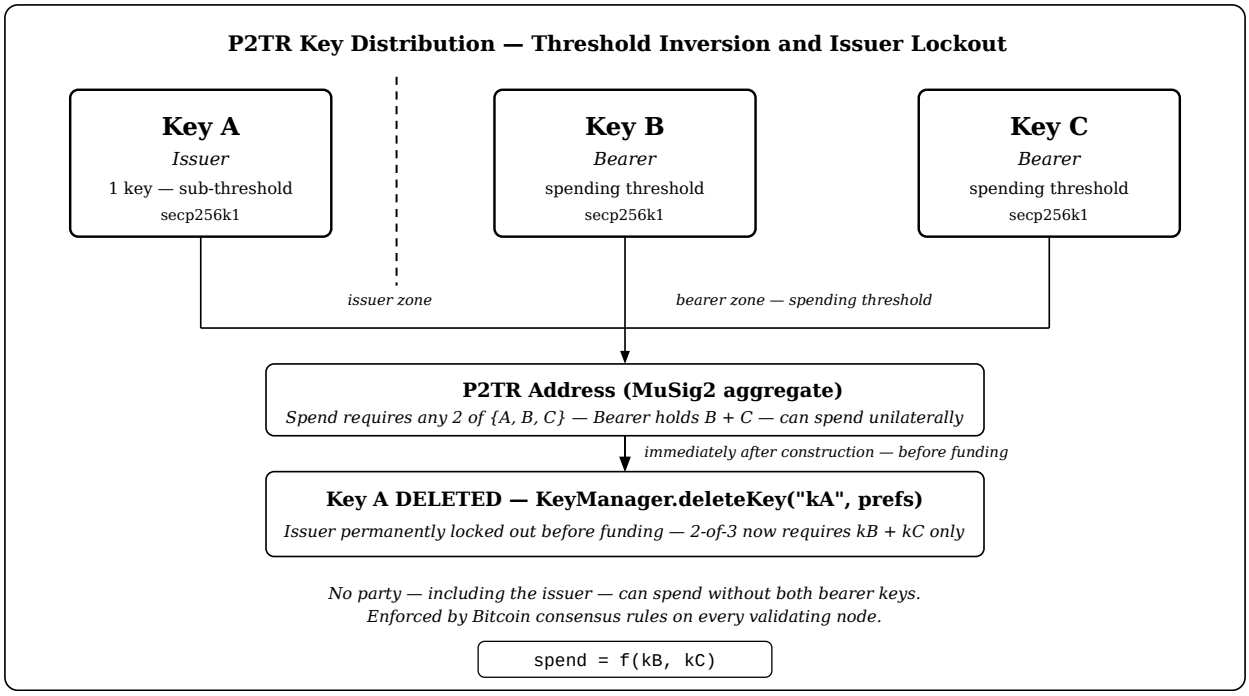

Figure 1: Key distribution architecture showing the threshold inversion. The instrument holder controls Keys B and C (spending threshold). The issuer holds only Key A (sub-threshold), deleted immediately after address construction before any Bitcoin is sent to the address.

3.2 Key Generation

All three keypairs are generated using a platform cryptographic random number generator with rejection sampling to ensure the generated scalar is strictly below the secp256k1 group order n. Keys exist as byte arrays in application memory during generation before being encrypted and written to hardware-backed storage.

Private keys are stored in hardware-backed encrypted storage using AES-256-SIV for key encryp-

tion and AES-256-GCM for value encryption. The wrapping key is held in a hardware security module and never leaves it. Access is gated by biometric authentication with a bounded session validity window. Private keys are never written to the plain local database; only the public keys K_A, K_B, and K_C appear in plain storage for receipt verification purposes.

All private key byte arrays are zeroed using `sodium_memzero()` immediately after use. This function uses platform-specific memory barrier instructions to prevent compiler elision of the zeroing operation. Private key objects are scoped to short-lived closures and never returned to callers.[1]

3.3 Buyer-Generated Key Issuance

If the issuer generates all three keypairs, the issuer has access to Keys B and C at generation time. A compromised or malicious issuer could retain copies and later spend outstanding instruments. This attack is closed by having the buyer generate Keys B and C on their own device. Only the public keys K_B and K_C are transmitted to the issuer. The issuer generates Key A independently, constructs the P2TR address from K_A, K_B, and K_C, and funds it. Recovering a private key from its corresponding public key requires solving the elliptic curve discrete logarithm problem over secp256k1, for which no efficient algorithm is known.

3.4 Transfer Mechanism

Transfer begins when the sender selects one or more instruments and the application marks them with a `pending_transfer` status in local storage. This prevents the same instrument from being presented for payment again during the transfer window. If the transfer is cancelled or the application terminates before the receiver accepts, a revert operation restores the instrument to active status. No keys have been deleted and no funds are at risk.

In the online path, the sender seals the instrument payload to the receiver's X25519 public key and publishes it as a Nostr kind 1 event. The receiver's device subscribes to the relay channel, decrypts the bundle on arrival, and verifies the UTXO. Delivery requires only that both parties can reach at least one common relay. A physical QR transfer path serves offline and in-person contexts: the receiver scans a QR code containing the sealed payload directly, with no relay dependency. The two paths share the same NaCl sealed-box construction and differ only in transport.

Keys are deleted by one of three paths depending on transfer mode. In the online path, keys are deleted only after the Nostr receipt proof passes the NC1 gate described in Section 5.2. In the offline P2P path, keys are deleted after the sender taps an explicit confirmation button; `kB` and `kC` are removed from hardware-backed encrypted storage and the instrument is marked transferred. On the merchant receive side, key deletion does not occur on the merchant's device; the merchant receives the keys and sweeps on network recovery.

Once the NC1 gate is passed and keys are deleted, the transfer is irreversible. The previous holder, no longer controlling both Keys B and C, cannot produce a valid spending signature. The new holder, controlling B and C, can spend unilaterally at any time.

Settlement occurs when the holder broadcasts a sweep transaction. In the P2TR key-path model, the holder produces a single aggregate Schnorr signature using Keys B and C via BIP-327 MuSig2. Every node on the network validates this independently. The sweep is the second and final on-chain event in the instrument's lifetime.

[1] The proof-of-concept uses `SecureRandom.getInstanceStrong()` for entropy on Android, with an Android Keystore wrapping key backed by StrongBox (Titan M2) or TEE. A known JVM limitation: bitcoin-kmp's `PrivateKey.value` is an immutable `ByteVector` that cannot be zeroed; its lifetime is minimised by scoping.

3.5 Verification and Attack Window

A recipient verifies an instrument by confirming the UTXO exists and carries the stated satoshi value; confirming the P2TR address encodes the expected MuSig2 aggregate public key derived from K_B and K_C with the script tree root; confirming Key A appears in the script tree and the holder controls both remaining keys; confirming the timelock expiry, if present, meets the minimum standard of Section 6; and querying nodes for unconfirmed spending transactions referencing the instrument address.

Verification queries two geographically independent nodes in parallel. UTXO presence, balance, and mempool state are cross-checked between both endpoints. If the two nodes disagree on UTXO existence, the instrument is rejected outright. For instruments at or above a configurable value threshold (50,000 sat in the current implementation), dual-node agreement is mandatory and single-node degradation is not permitted. Below this threshold, verification degrades to single-node with a warning when the second endpoint is unreachable.

In the digital transfer path, private key bytes are zeroed via `sodium_memzero()` immediately after the NaCl decryption operation completes. The window between decryption and zeroing is bounded by the NaCl open operation, typically under one millisecond on current hardware. An attacker who intercepts the sealed payload cannot extract the keys without the recipient's X25519 private key. The primary defence against the pre-broadcast sweep race is immediate sweep by the recipient, reinforced by dual-node mempool scanning that detects competing spend transactions visible to either verification endpoint.

4 Off-Chain Transfer Economics

A physical note changes hands many times between issuance and final settlement. Each transfer is free, instantaneous, and requires no network connection. Two events involve the issuing party: funding and final redemption. Between them the instrument circulates at no cost.

An MBI instrument shares this economic structure. Two on-chain transactions bound the instrument's lifetime. Between them it circulates through key handover at zero cost, zero latency, and with no network requirement. An instrument that has transferred forty times carries the same value as one that has transferred once, less the eventual sweep fee. This differs from all prior digital payment mechanisms, in which settlement costs accumulate with each transfer.

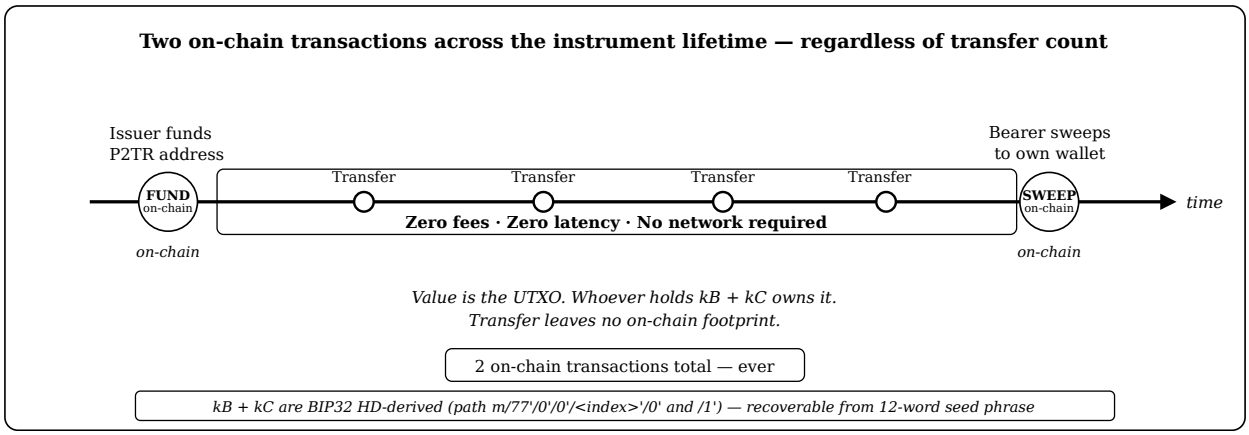

Figure 2: Instrument lifecycle. Two on-chain transactions bound the instrument's entire existence. Between funding and settlement, any number of free, instant, offline transfers occur with no on-chain footprint.

4.1 Denomination and Fee Economics

Instruments are denominated in satoshis. Fiat denomination is not recommended because Bitcoin price movement changes the real value of outstanding instruments over time. Batch issuance funds many instruments in a single transaction with one input and multiple outputs. Transaction fees scale primarily with byte size rather than output count, so per-instrument issuance cost at batch scale is low. At 5 sat/vbyte, a batch of 100 instruments at 10,000 sat costs approximately $0.93 in total fees, or 0.19% of float value. A P2TR key-path sweep transaction is approximately 111 virtual bytes; at the same fee rate the sweep costs approximately $0.055 regardless of denomination. This establishes a practical denomination floor near $1 at normal fee rates. Recipients who accumulate multiple instruments and batch their sweeps reduce the per-instrument settlement cost proportionally.

Table 1: Fee comparison across payment methods for 100 × $5 payments.

Payment method	Per-transfer fee	100 × $5 total
MBI (batch issue + batch sweep)	$0.00	∼$1.49
Lightning Network	∼$0.001 routing	∼$3.00
Bitcoin on-chain	∼$0.50 per tx	∼$100.00
Visa / Mastercard	2.9% + $0.30 per tx	∼$44.50

5 Digital Transfer and MBI-PRP

5.1 QR Payload Construction

For digital transfer, the sender constructs a payload containing the P2TR address, the satoshi value, the issuer public key K_A, and the spending private keys k_B and k_C. The payload is compressed and sealed using the recipient's X25519 public key. This function performs an ephemeral X25519 key exchange and encrypts under XSalsa20-Poly1305 with the resulting shared secret. The sender's identity is not embedded in the ciphertext. Only the holder of the corresponding X25519 private key can open it. The sealed ciphertext is placed in a JSON envelope alongside a 32-byte random nonce, a Nostr routing identifier, and the sender's X25519 public key. This envelope is encoded as a QR code. A party who photographs the QR code obtains only ciphertext; the spending keys are not recoverable without the recipient's private key.

5.2 Nostr Receipt and the Transfer Trust Model

The sender cannot delete spending keys on QR display without risking fund loss if the receiver never scans. The sender cannot retain keys indefinitely without maintaining a double-spend capability. The protocol requires a mechanism by which the sender can confirm the receiver holds the spending keys before deleting their own copies.

On scanning and decrypting the QR payload, the receiver's device automatically computes an ECDSA signature over the nonce using k_B: $\text{sig} = \text{sign}(k_B, \text{nonce})$. This signature can only be produced by the holder of k_B. The device constructs a receipt containing the signature, the instrument address, the routing identifier, and the public key K_B; encrypts it to the sender's X25519 public key using `crypto_box_seal`; and publishes it as a Nostr text note of kind 1. The receipt is published as a standard Nostr event, routed to the sender via a public-key subscription filter. The Nostr event is signed with a throwaway secp256k1 keypair that is zeroed immediately after the Schnorr signature is computed. Relay selection is a configurable deployment parameter.

On receiving the encrypted receipt, the sender decrypts it and verifies the ECDSA signature against K_B. The NC1 gate is then applied: key deletion proceeds only if the instrument address is present

in `onChainVerifiedAddresses`, a set populated only after independent UTXO confirmation by the receiver's device. This gate prevents an adversary from triggering key deletion by publishing a receipt for an unfunded address. When both conditions are satisfied, the application removes k_B and k_C from hardware-backed encrypted storage.

Trust Model for Sender Key Deletion

The receipt proves one thing: the receiver holds k_B. Only the holder of k_B can produce a valid ECDSA signature over the per-payment nonce. It does not prove the sender has deleted their copies of k_B and k_C. In a software-only deployment, sender deletion is performed by the application and assumed by the protocol, not enforced by an external mechanism.

Two partial mitigations bound the practical risk of sender non-deletion. First, the structural argument: a sender retaining only one of k_B or k_C cannot produce a valid sweep transaction, since both keys are required to meet the 2-of-3 threshold with k_A gone. A sender wishing to double-spend must therefore retain *both* k_B and k_C, then use them after a receiver who possesses a timestamped cryptographic proof of possession of k_B has been issued value. This constitutes detectable on-chain fraud against a party holding verifiable evidence.

Second, the economic argument: the sender has already received value in exchange for the instrument. Double-spending by retaining both keys, then sweeping the UTXO, recovers the instrument value while leaving a cryptographic trail linking the act to the sender's public key and device. At low denominations the attack is irrational given this exposure.

For deployments where the software-only trust model is insufficient, the TEE upgrade path addresses the gap directly. Spending keys generated and stored inside a qualifying hardware Secure Enclave cannot be exported by application code. The enclave can atomically mark the keys as transferred and refuse subsequent signing operations, producing a signed attestation certificate the receiver can verify. This converts assumed deletion into hardware-enforced deletion with an auditable attestation chain. On qualifying Android devices at API level 31 and above, StrongBox hardware security modules already support secp256k1 key generation within the TEE. Apple Secure Enclave does not currently support secp256k1; this is a Bitcoin protocol constraint that applies to all Bitcoin applications on that platform. ARM Confidential Compute Architecture on ARMv9 processors provides a platform-neutral path to enclave-isolated secp256k1 operations. The tradeoff is that every participant must use a device with a qualifying TEE, and all parties must trust that attestation chain. In a consumer peer-to-peer context this is a significant deployment constraint; the TEE model fits issuer-managed or institutional deployments more readily. For low-denomination everyday transfers the economic argument operates as the primary trust mechanism, which is the appropriate model for the use cases this protocol targets.

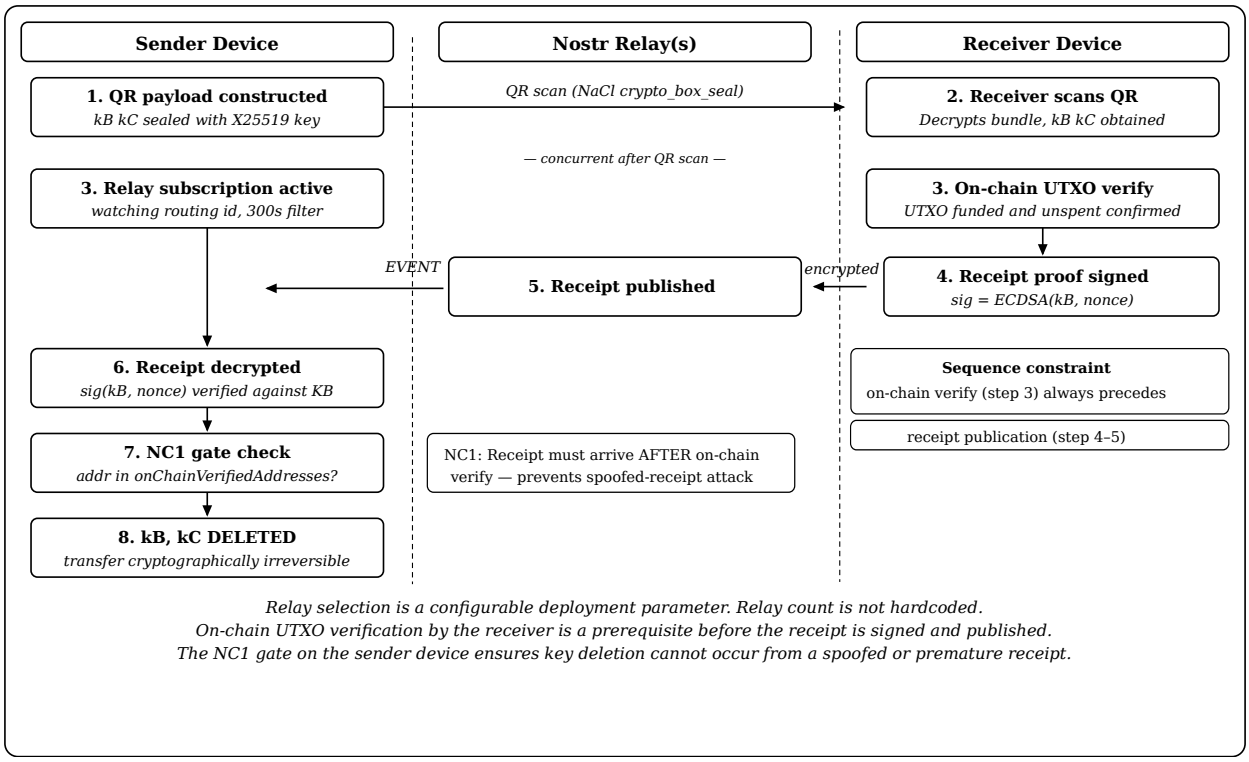

Figure 3: MBI-PRP payment flow and Nostr receipt mechanism. A single QR scan on the receiver's device triggers the full automatic sequence. The sender applies the NC1 gate before irrevocably deleting k_B and k_C. Relay selection is a configurable deployment parameter.

5.3 MBI Payment Request Protocol

MBI-PRP is a payment request standard for app-to-app and app-to-website payments using MBI instruments. No prior relationship, account, or registration between payer and payee is required. A payment request contains the payee's NaCl public key, the requested satoshi amount, a one-time request identifier, an expiry timestamp, and a callback endpoint. The payer selects instruments covering the requested amount, seals the spending keys to the payee's public key, and transmits the bundle. The payee decrypts, verifies the UTXO balance and script structure, and returns confirmation.

The protocol is transport-agnostic by design. HTTPS serves web checkout. QR codes serve mobile scan payments. Nostr relay delivery serves online peer-to-peer transfers. The full specification includes NFC NDEF for in-person tap payments; this transport binding is specified design and is scoped as future work. No existing standardised payment protocol occupies the position of requiring no account, operating fully offline, and incurring zero fee per payment.

Table 2: Comparison of standardised payment protocols.

Standard	Account required	Works offline	Fee per payment	Open
Apple Pay	Yes	No	0.15%	No
Lightning BOLT11	Yes	No	Routing fee	Yes
BIP 21 URI	Yes	No	On-chain fee	Yes
MBI-PRP (this paper)	**No**	**Yes**	**Zero**	**Yes**

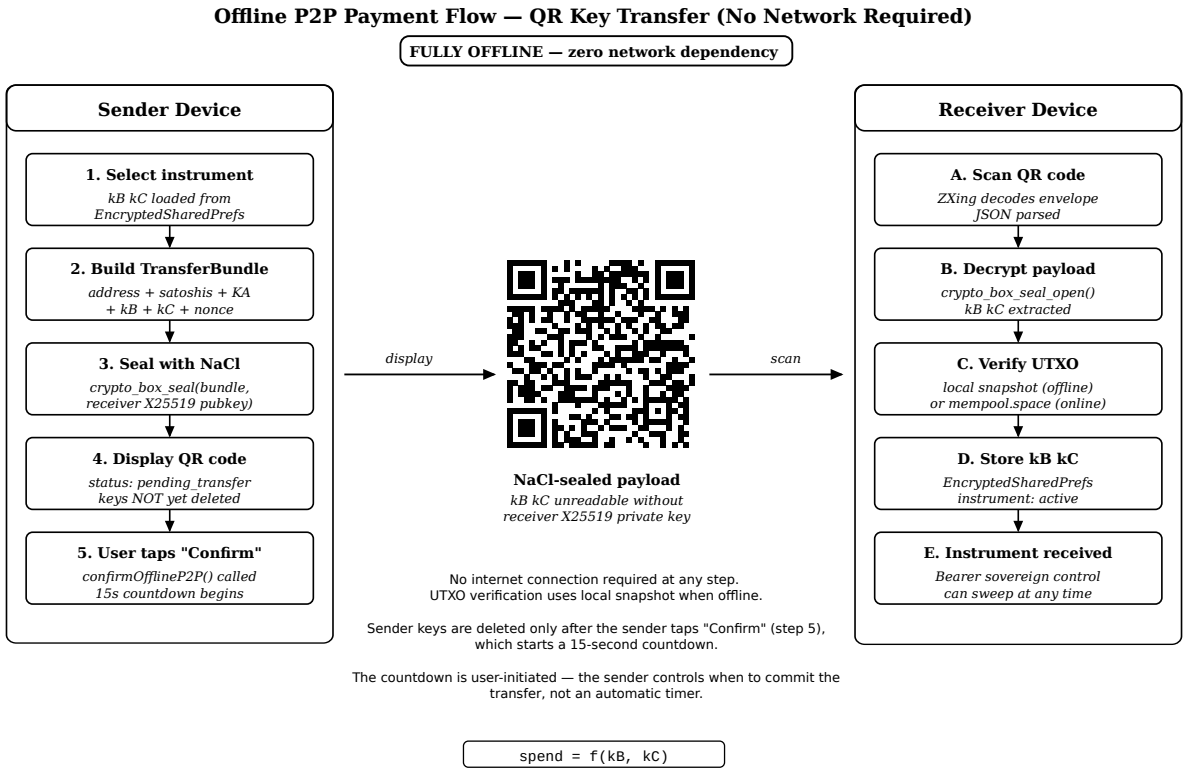

Figure 4: Offline P2P transfer protocol. Scan 1 obtains the receiver's X25519 public key. Scan 2 transfers the NaCl-sealed instrument payload. Key deletion follows one of three paths: immediate on verified Nostr receipt (online); after explicit user confirmation and 15-second countdown (P2P offline); or not at all on the merchant side, where the receiver holds the spending keys and sweeps on network recovery.

6 Timelock-Based Issuer Recovery

An instrument whose spending keys are permanently lost is permanently unspendable. To address this, the Tapscript tree may optionally include a recovery leaf using `OP_CHECKLOCKTIMEVERIFY`. The script tree encodes two spending paths: the key path using the MuSig2 aggregate of K_B and K_C, accessible at any block height, and a Tapscript recovery leaf requiring only Key A, accessible after a specified block height. The recovery leaf commitment is embedded in the Taproot tweak and can be verified independently by any party who holds all three public keys. Instruments with no recovery leaf are valid and carry no expiry.

Timelock recovery is a specified design feature and is scoped as future work. The current proof-of-concept constructs P2TR addresses with a script tree containing the 2-of-3 `CHECKSIGADD` leaf but no `CLTV` recovery leaf.

6.1 Short Timelock Attack and Minimum Standard

A sender who encodes a short timelock can transfer the instrument, allow the recipient to accept it in good faith, wait for expiry, and recover the funds using Key A. Verification software must enforce a minimum remaining timelock before accepting an instrument. The minimum is a configurable parameter dependent on instrument denomination and use case. For low-denomination instruments in everyday payment contexts, a minimum of 4,320 blocks (approximately one month) is appropriate. For higher-value instruments or institutional applications, longer minimums are warranted. Instruments with fewer blocks remaining than the configured minimum must be rejected regardless of funded state.

7 Adversarial Security Analysis

We consider a technically capable adversary with full knowledge of the protocol. Twelve attacks are identified. Nine are closed by the protocol design. Two require true hardware Secure Enclave key generation and are noted as open in the current proof-of-concept. One, sender key retention after transfer, is mitigated by structural and economic arguments in the software model and closed only by the TEE upgrade path. The trust model implications of this last point are discussed separately in Section 5.2.

7.1 Critical Attacks

Malicious issuer records Keys B and C. If the issuer generates all three keypairs, the issuer has access to Keys B and C at generation time. Closed by buyer-generated key issuance as described in Section 3.4.

Copied key multi-sale. A holder copies Keys B and C before transfer and sells the same instrument to multiple parties. On qualifying devices the proof-of-concept implements hardware-backed key storage via the platform hardware security module, which confines key material within the secure element. Full Secure Enclave enforcement with per-operation attestation remains a further hardening step for production deployment. The primary practical mitigation for merchant receipts is immediate sweep, which reduces the exploitable window to the mempool propagation time of the sweep transaction.

7.2 High-Exploitability Attacks

Sender key retention after transfer. The sender retains k_B and k_C after the Nostr receipt is verified and proceeds to sweep the UTXO, defrauding the receiver. In the software model this is mitigated structurally (both keys must be retained; retaining only one is insufficient to spend) and economically (the act is traceable and the receiver holds a cryptographic proof of possession of k_B). Full cryptographic closure requires TEE enforcement as described in Section 5.2. Status: partially mitigated in software; closed with Secure Enclave.

Pre-broadcast sweep. The attacker pre-constructs a sweep transaction and broadcasts it during the recipient's verification window. Bitcoin mempool propagation takes 30 to 60 seconds to reach a supermajority of nodes. Primary closure: the recipient sweeps within seconds of accepting the instrument. Secondary closure: the verification protocol queries two geographically independent nodes in parallel for mempool activity referencing the instrument address. A competing spend visible to either node triggers rejection before the recipient accepts the instrument.

NC1: forged receipt triggering premature key deletion. An adversary publishes a Nostr receipt for an instrument whose on-chain verification has not yet completed. Closed by the NC1 gate: key deletion requires the instrument address to be present in `onChainVerifiedAddresses`, populated only after independent UTXO confirmation by the receiver's device. Both the cryptographic receipt and on-chain confirmation must be satisfied before deletion proceeds.

Memory extraction. A rooted device or modified application extracts key bytes from memory. On qualifying devices the proof-of-concept stores keys within the platform hardware security module such that key material does not exist in application memory. Devices without a dedicated secure element fall back to TEE-backed storage. Per-operation Secure Enclave attestation with transfer marking remains a further hardening step.

Timing attack on merchant verification. Closed by re-querying immediately before release and by immediate sweep.

Network partition and eclipse attack. Closed by querying two independent nodes simultaneously. An attacker must compromise or partition both endpoints to prevent detection of a

competing spend. TLS on both connections prevents response modification in transit.

MITM public key substitution. Closed by TLS on all network transport.

Unfunded instrument. Closed by UTXO verification against a confirmed funded state before accepting the instrument.

Receipt replay. Closed by the per-payment 32-byte random nonce; a receipt signature produced over one nonce is invalid for any other.

Table 3: Attack surface summary.

Attack	Severity	Status
Malicious issuer copies B+C	Critical	Closed: buyer-gen. keys
Copied key multi-sale	Critical	Requires Secure Enclave
Sender retains keys after transfer	High	Partial: structural + economic; closed with TEE
Mempool race / pre-broadcast sweep	High	Closed: immediate sweep + dual-node mempool scan
NC1: forged receipt / premature deletion	High	Closed: NC1 gate
Memory extraction	High	Requires Secure Enclave
Timing / partition / MITM	Moderate	Closed: dual-node + TLS
Fake instrument / receipt replay	Low	All Closed

8 Implementation Considerations

8.1 Software Stack

Full implementation requires P2TR address generation with BIP-327 MuSig2 key aggregation and BIP-341 Tapscript tree construction, available in bitcoin-kmp, secp256k1-kmp, libsecp256k1, and Bitcoin Core. BIP-341 sighash computation and Schnorr signing produce a single 64-byte witness per input. `OP_CLTV` script encoding per BIP 65 supports optional timelock recovery within a Tapscript leaf. libsodium provides the NaCl primitives used in digital transfer. A platform hardware security module provides hardware-backed wrapping key storage.

8.2 Key Storage in the Proof-of-Concept

Spending private keys k_B and k_C are stored in hardware-backed encrypted storage using AES-256-SIV for key encryption and AES-256-GCM for value encryption. The wrapping key is held in a hardware security module and never leaves it. Access is gated by biometric authentication with a bounded session validity window. Private keys are not stored in the plain local database; only the public keys K_A, K_B, and K_C appear there for receipt verification. All private key byte arrays are zeroed via `sodium_memzero()` immediately after use. Key generation uses a platform cryptographic random number generator with rejection sampling. The AES-256-GCM wrapping layer retains 128-bit effective security against an adversary with access to a quantum computer running Grover's algorithm.

The full production architecture specifies key generation inside a hardware Secure Enclave with a per-operation biometric gate, such that private key bytes never exist in application memory. Transition to this model requires hardware support for secp256k1 operations within the secure element.

Table 4: Platform support for hardware-backed secp256k1 key storage.

Platform	HSM / Secure Element	secp256k1 in TEE	Attestation
Android (API 31+, StrongBox)	Titan M2 / Knox	Yes	Key Attestation
Apple (iOS 16+)	Secure Enclave	No (P-256 only)	App Attest
ARM CCA (ARMv9)	Realm isolation	Implementation-dependent	CCA attestation
Server TEE (SGX / SEV / Phala)	Enclave memory	Yes (libsecp256k1)	Remote attestation

Apple CryptoKit supports P-256 and post-quantum algorithms (ML-KEM, ML-DSA as of iOS 26) in the Secure Enclave but does not support secp256k1. The Bitcoin spending operations require secp256k1 regardless of platform, which is a Bitcoin protocol constraint, not a platform limitation.

8.3 Bounded UTXO Set and Offline Verification

The full protocol specification describes a bounded UTXO set indexed from the mainnet inception block: the first Bitcoin mainnet block in which a funded MBI instrument appears at launch. MBI instruments are identifiable by script template and issuer public key, so the verification application indexes only matching UTXOs from that block forward. No prior blockchain history is relevant. Swept instruments are removed from the index. The set grows in proportion to active instrument circulation, not cumulative issuance.

At 100,000 active instruments the set occupies approximately 15 MB. At 1,000,000 active instruments approximately 150 MB. Initial synchronisation uses an issuer-signed snapshot verified against block headers via SPV. Ongoing synchronisation uses BIP 157/158 compact block filters to identify relevant blocks without downloading full block data. A device with a synchronised snapshot can verify any previously indexed instrument without network access.

This offline verification capability is scoped as future work. The current proof-of-concept verifies instruments by querying two independent mempool.space endpoints over HTTPS with dual-node cross-checking (Section 3.5) and does not implement local UTXO indexing. Full offline verification depends on this capability and on the NFC NDEF transport binding described in Section 5.3.

9 Comparison with Related Systems

Prior off-chain threshold and custodial systems achieve some of the relevant properties but not all simultaneously. Ecash systems using blind signatures allow off-chain transfer at zero per-transfer cost but require the mint as a trusted participant at redemption. Payment channel networks allow off-chain settlement at low fees but require online counterparties and channel liquidity. Physical single-key instruments allow off-chain transfer by key handover but depend on trust in the issuer's non-retention. Multisignature schemes that position the issuer as a co-signer preserve custodial dependency through settlement.

The present proposal achieves issuer lockout through the signature threshold itself. No coalition of issuer keys meets the spending threshold. The issuer is not required at settlement, is not required to be online, and cannot be compelled to act on the instrument. The issuer's key does not exist after address construction. This combination of properties has not been achieved by any prior system. The property is verifiable by any party who can derive the P2TR address from the three public keys and confirm the threshold configuration.

Table 5: Comparison with related systems.

System	Issuer lockout	Off-chain transfer	Fee	No trust
Single-key instrument	No	Yes	None	No
Write-once hardware	Partial (hw)	Yes	None	Partial
Issuer co-sign multisig	No	Yes	None	No
Ecash (blind sig)	Yes (blind sig)	Yes	None	No (mint)
Lightning Network	N/A	Yes	Routing	No (LSP)
MBI (this paper)	**Yes (by threshold)**	**Yes**	**None**	**Yes**

10 Privacy Analysis

At issuance and settlement, MBI instruments are visible on-chain. The funding transaction reveals the P2TR address and denomination. Between the funding and sweep transactions the instrument leaves no on-chain trace. Key handover between sender and recipient produces no blockchain record.

When a recipient queries an external endpoint to verify an instrument, the instrument address and the recipient's network address are disclosed to the endpoint operator. Verification against a local UTXO snapshot eliminates this exposure. Fixed denominations allow chain analysts to identify probable MBI outputs by amount. Issuers can mitigate this by adding a small random satoshi offset at funding time, varying each instrument's value within a defined range above the base denomination.

The P2TR address type with BIP-327 MuSig2 key aggregation removes the on-chain multisig fingerprint. Keys A, B, and C aggregate into a single internal public key. The resulting address is indistinguishable from a standard single-signature P2TR address. The holder produces a single aggregate Schnorr signature using Keys B and C. No chain analyst can determine from the sweep transaction that a threshold scheme was involved. This property requires no Bitcoin protocol changes beyond Taproot, activated in November 2021 under BIP 340 through 342.

11 Quantum Safety Evaluation

The asymmetric primitives used in this system — secp256k1 ECDSA, secp256k1 Schnorr, and X25519 — are vulnerable to Shor's algorithm on a cryptographically relevant quantum computer. This vulnerability is common to Bitcoin generally and is not specific to this protocol. No quantum computer capable of breaking 256-bit elliptic curve cryptography exists or is considered imminent; published estimates place such capability at a minimum of ten or more years.

The symmetric layer is not vulnerable in the same sense. AES-256-GCM, used in the hardware-backed wrapping key, retains 128-bit effective security against Grover's algorithm. SHA-256, used in Bitcoin transaction hashing and Nostr event identifiers, retains 128-bit collision resistance. These components do not require modification.

The asymmetric components are isolated in two classes. `ReceiptProof` handles ECDSA signing and verification for the Nostr receipt mechanism. `NaclIdentity` handles X25519 key exchange for payload encryption.

The receipt layer has been upgraded to include ML-DSA-44 hybrid signatures per NIST FIPS 204 [20]. Each receipt now carries both an ECDSA signature and an ML-DSA-44 signature over the same per-payment nonce. A receiver verifying both obtains post-quantum assurance on the receipt proof even while the underlying Bitcoin layer remains classically secured. The ML-DSA implementation uses a deterministic key derivation from the existing k_B seed material, avoiding any new key distri-

bution requirement. During the transition period, ML-DSA verification failure is logged but does not reject the receipt, allowing receivers running older software to accept receipts from upgraded senders.

The payload encryption layer remains X25519. Replacing it with ML-KEM-768 per NIST FIPS 203 [19] constitutes the next post-quantum upgrade. ML-KEM ciphertexts are approximately 1,088 bytes, which is compatible with the Nostr relay transport but exceeds practical QR code capacity at current error correction levels. The QR path will retain classical encryption until a density-compatible post-quantum KEM is available or until the QR path is supplemented by NFC or WiFi Direct transport. The Bitcoin multisig layer depends on a network-level upgrade that applies to all Bitcoin applications equally.

12 Generalisation to Other Settlement Layers

The mechanism described in this paper requires three properties of the underlying settlement layer: threshold multisignature spending enforced by consensus rules; a verifiable funded state equivalent to a UTXO; and off-chain key transmission capability. Bitcoin-derived UTXO-based networks that support Taproot or equivalent threshold signature constructions can adopt this mechanism directly. Stablecoin-denominated instruments on Layer 2 networks are a commercially significant application: a fixed-value instrument eliminates the denomination volatility while retaining all threshold instrument properties.

The threshold inversion principle applies to any context in which post-issuance issuer control is a structural problem. Structured settlement instruments, commercial paper, trade finance, and wholesale CBDC designs share the same limitation in their current electronic form: transfer requires a registry update and the issuing institution retains the ability to freeze, reassign, or cancel. A threshold-configured instrument restores direct transferability at the cryptographic layer. Identity verification at issuance addresses the regulatory requirements relevant to each deployment context. For regulated deployments where blockchain settlement is not appropriate, the same threshold configuration is implementable using hardware security modules as key custodians with settlement on a permissioned ledger.

13 Changes from Version 1

Version 2 replaces P2SH address construction with Pay-to-Taproot (BIP-327 MuSig2 key aggregation, BIP-341 Tapscript leaves); adds parallel dual-node UTXO verification with mandatory agreement above a configurable threshold; adds ML-DSA-44 hybrid receipt signatures (FIPS 204) alongside ECDSA; deploys the proof-of-concept on Bitcoin Mainnet; and makes Nostr relay delivery the primary online transfer path. Version 1 P2SH instruments are not compatible with version 2 logic. No version 1 instruments were issued on mainnet.

14 Conclusion

We have proposed a system for transferring Bitcoin value without on-chain settlement at each transfer, without custodial dependency on the issuer, and without network connectivity at payment time. The central contribution is a specific inversion of the key distribution used in all prior multisignature schemes: the holder controls the full spending threshold and the issuer holds a sub-threshold key that is arithmetically insufficient to authorise any transaction. The issuer's key is deleted before the instrument is funded. No copy of it exists during the instrument's funded lifetime.

The Nostr receipt mechanism provides the sender with cryptographic proof that the receiver holds the spending keys. The receipt does not prove the sender deleted their copies. The NC1 gate ensures key deletion cannot be triggered by a forged receipt for an unfunded instrument. The `pending_transfer` intermediate state allows cancellation without fund loss up to the point of irrevocable key deletion. In the software model, the sender's deletion is assumed and supported by structural and economic arguments; the TEE upgrade path converts this to hardware-enforced assurance for deployments that require it. Two on-chain transactions bound the instrument's lifetime regardless of transfer count. Between them it circulates as instrument value with no fees, no latency, and no network requirement.

A working implementation on Bitcoin Mainnet demonstrates these properties. A batch funding transaction confirmed in block 944,598 on 2026-04-11:

`7190ed916437f5abd9a7b9e141bc60bcefd9070e5b1f92797bae4b2ed99879c7`

demonstrates batch issuance of multiple denominated P2TR instruments in a single on-chain transaction. A complete instrument lifecycle—funding, transfer, and key-path Schnorr sweep—is recorded at address:

`bc1qe3qtx22rvkx4pqz7s6yvkstl8fjll4007e7nza`

The funding transaction confirmed in block 944,597 and the sweep transaction confirmed in block 944,598. Both transactions are verifiable on any Bitcoin Mainnet block explorer. The Tapscript leaf constructions and timelock recovery paths have been verified by two independent implementations (Kotlin and Python) producing identical addresses and signatures, and by mainnet consensus acceptance of the sweep transactions.

Directions for further work include implementation of the bounded UTXO set and offline verification described in Section 8.3; hardware Secure Enclave integration for enforced key deletion with attestation; NFC NDEF transport for tap payments; and the ML-KEM post-quantum upgrade for payload encryption. The system requires no changes to the Bitcoin protocol and no trust in any party beyond the mathematics of threshold signatures and the consensus rules that enforce them.

鍵音ひびく
扉の影集う
会の朝霧

References

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

[2] David Chaum. "Blind Signatures for Untraceable Payments". In: *Advances in Cryptology — CRYPTO 1982*. Springer, 1983, pp. 199–203.

[3] Peter W. Shor. "Algorithms for Quantum Computation: Discrete Logarithms and Factoring". In: *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. IEEE, 1994, pp. 124–134.

[4] Matthew Green and Giuseppe Ateniese. "Identity-Based Proxy Re-Encryption". In: *Applied Cryptography and Network Security — ACNS 2007*. Springer, 2007, pp. 288–306.

[5] Gavin Andresen. *BIP 16: Pay to Script Hash*. Bitcoin Improvement Proposal. 2012.

[6] Daniel J. Bernstein, Tanja Lange, and Peter Schwabe. *NaCl: Networking and Cryptography Library*. `https://nacl.cr.yp.to`. 2012.

[7] Nils Schneider and Matt Bengsson. *BIP 21: URI Scheme*. Bitcoin Improvement Proposal. 2012.

[8] Peter Todd. *BIP 65: OP_CHECKLOCKTIMEVERIFY*. Bitcoin Improvement Proposal. 2014.

[9] Thomas Voegtlin. *BIP 67: Deterministic Pay-to-script-hash multi-signature addresses through public key sorting*. Bitcoin Improvement Proposal. 2015.

[10] Joseph Poon and Thaddeus Dryja. *The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments*. `https://lightning.network/lightning-network-paper.pdf`. 2016.

[11] Olaoluwa Osuntokun, Alex Akselrod, and Jim Lamarque. *BIP 157/158: Client-Side Block Filtering*. Bitcoin Improvement Proposal. 2017.

[12] Andrew Towns et al. *BIP 174: Partially Signed Bitcoin Transaction Format*. Bitcoin Improvement Proposal. 2017.

[13] Pieter Wuille, Jonas Nick, and Andrew Poelstra. *BIP 340–342: Schnorr Signatures, Taproot, Tapscript*. Bitcoin Improvement Proposal. 2020.

[14] Jonas Nick, Tim Ruffing, and Yannick Seurin. "MuSig2: Simple Two-Round Schnorr Multi-Signatures". In: *Advances in Cryptology — CRYPTO 2021*. Springer, 2021.

[15] Cashu Contributors. *Cashu: Chaumian Ecash for Bitcoin*. `https://cashu.space`. 2022.

[16] Fedimint Contributors. *Fedimint: A Federated E-Cash Mint Protocol*. `https://fedimint.org`. 2022.

[17] Nostr Protocol. *NIP-01: Basic Protocol Flow*. `https://github.com/nostr-protocol/nostr`. 2022.

[18] Jonas Nick, Tim Ruffing, and Yannick Seurin. *BIP 327: MuSig2 for BIP 340-Compatible Multi-Signatures*. Bitcoin Improvement Proposal. 2023.

[19] National Institute of Standards and Technology. *FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard (ML-KEM)*. `https://csrc.nist.gov/pubs/fips/203/final`. 2024.

[20] National Institute of Standards and Technology. *FIPS 204: Module-Lattice-Based Digital Signature Standard (ML-DSA)*. `https://csrc.nist.gov/pubs/fips/204/final`. 2024.