# B2: Anonymous Payments on Solana via a Program-Owned Shielded Vault

Technical Paper · Version 0.1

November 2026

**Abstract**

B2 is a payment protocol on Solana that hides the sender, the recipient, the amount and the asset of a transfer. It is built as a zero-knowledge application on top of the default ring of Solana Privacy Rings (SPR), a UTXO-based confidential transfer system whose shielded pool program (SPP) hides amounts and assets but publishes the owner of every UTXO. B2 closes that gap with a program-owned vault: all value held through B2 is owned, at the SPR layer, by a single program-derived identity, and per-user ownership is enforced one layer above, inside B2's own circuits. Every private B2 transaction therefore carries two Groth16 proofs over BN254: an SPP proof that the confidential state transition of the pool is valid, and a vault proof that a B2 user is authorised to perform it, both verified on chain in the same Solana transaction. Proofs are generated on the user's device; a fee relay submits transactions so that no account of the user signs a private transfer; and the wallet rebuilds the commitment and nullifier trees locally from a user-chosen RPC endpoint, without trusting an indexer. We specify the construction, state its security properties and their assumptions, report measured parameters of the mainnet deployment ($128 + 192$ bytes of proof per transfer, $\approx 2.4 \times 10^5$ compute units, sub-two-second client-side proving), and outline the roadmap.

## Contents

**1 Introduction** **2**

**2 Preliminaries** **2**

**3 The substrate: Solana Privacy Rings** **3**

**4 Architecture** **4**

**5 Circuits** **5**

**6 On-chain verification and accounting** **6**

**7 Trusted setup** **7**

**8 Network layer: the fee relay** **7**

**9 Client** **7**

**10 Security** **8**

**11 Trust model and roadmap** **9**

**A Deployment** **9**

**B Parameters** **9**

# 1 Introduction

Account-based blockchains publish every transfer: sender, recipient, amount, asset and time. On Solana this record is permanent, globally replicated and trivially indexed, so a single address reuse links a user's entire financial history. Existing privacy mechanisms on Solana address parts of this problem. Token-2022 confidential transfers encrypt amounts under twisted ElGamal but leave both parties public. The default ring of Solana Privacy Rings (SPR) hides amounts *and* assets inside Poseidon commitments, but deliberately publishes the owner of every UTXO so that wallets can synchronise at RPC speed (Section 3).

B2 targets the remaining property, *anonymity*: an observer of the chain should learn that a B2 transfer took place, and nothing about who sent it, who received it, how much moved or which asset. B2 achieves this without deploying its own pool and without asking its users to join a separate anonymity set from scratch: it inherits SPR's data structures (a depth-32 Poseidon UTXO tree and a height-40 indexed nullifier tree), its confidential transfer circuit and its on-chain verifier, and adds a thin, separately proven ownership layer.

**Contributions.**

1. A *program-owned vault* construction that turns a confidential, owner-public UTXO system into an anonymous one by collapsing all public owners into one program-derived identity and re-establishing individual ownership in zero knowledge (Section 4).
2. A *dual-proof transaction* format in which an application proof and the pool proof are bound to the same private transaction hash and verified atomically (Section 5.1).
3. A client architecture with local proving, relay-based submission and indexer-free state reconstruction (Section 9).
4. A mainnet deployment with on-chain deposit limits and a precisely stated trust model (Sections 6, 10 and 11).

**Scope.** This paper describes version 0.1 of B2 as deployed on Solana mainnet in November 2026. Where B2 relies on SPR, we summarise the relevant parts of the SPR specification (crate version 0.10.0) rather than restating it in full.

# 2 Preliminaries

## 2.1 Notation

$\mathbb{F}_r$ denotes the scalar field of BN254, with

$$r = \texttt{0x30644e72e131a029b85045b68181585d2833e84879b9709143e1f593f0000001} \approx 2^{253.6}.$$

Elements of $\mathbb{F}_r$ are represented as 32-byte big-endian strings. $\mathbb{G}_1$, $\mathbb{G}_2$ are the order-$r$ subgroups of $E(\mathbb{F}_p)$ and $E'(\mathbb{F}_{p^2})$, $\mathbb{G}_T \subset \mathbb{F}^*_{p^{12}}$, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is the optimal Ate pairing. $x \xleftarrow{\$} S$ denotes uniform sampling; $a \,\|\, b$ concatenation; $[k]$ the set $\{0, \ldots, k-1\}$.

## 2.2 Hash functions

$\mathsf{H}$ denotes the Poseidon permutation-based hash over $\mathbb{F}_r$ as used by SPR, with arity determined by the number of arguments. Two derived constructions appear throughout:

- $\mathsf{hb}_N(b)$ commits to a fixed-length byte string $b \in \{0,1\}^{8N}$ by splitting it into 31-byte big-endian chunks $c_0, c_1, \ldots$, setting $h \leftarrow c_0$ and folding $h \leftarrow \mathsf{H}(h, c_i)$. Strings of at most 31 bytes map to their packed field value.
- $\mathsf{Sha256BE}(m)$ is SHA-256 of $m$ with the most significant byte cleared, interpreted as an element of $\mathbb{F}_r$.

$\mathsf{HashChain4}(e_0, \ldots, e_{L-1})$ folds left to right three elements per call of the 4-input Poseidon permutation, zero-padding the final group; $\mathsf{RightHashChain4}$ folds right to left. Neither is length-prefixed, so each is used only over inputs whose length is fixed by the compiled circuit.

### 2.3 Groth16

For an arithmetic circuit $C$ compiled to a rank-1 constraint system (R1CS) with $m$ constraints and $\ell$ public inputs, Groth16 [1] provides a non-interactive argument with a proof

$$\pi = (A, B, C) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1$$

of three group elements, independent of $m$. Given a verifying key $\mathsf{vk} = ([\alpha]_1, [\beta]_2, [\gamma]_2, [\delta]_2, \{[\mathrm{IC}_i]_1\}_{i=0}^{\ell})$ and public inputs $a_1, \ldots, a_\ell \in \mathbb{F}_r$, the verifier accepts iff

$$e(A, B) \;=\; e([\alpha]_1, [\beta]_2) \cdot e\Big(\textstyle\sum_{i=0}^{\ell} a_i [\mathrm{IC}_i]_1,\; [\gamma]_2\Big) \cdot e(C, [\delta]_2), \qquad a_0 = 1.$$

This is a single multi-pairing equation. Groth16 is knowledge-sound in the generic group model and perfectly zero-knowledge for any well-formed common reference string (CRS). Its soundness depends on the secrecy of the setup randomness $(\alpha, \beta, \gamma, \delta, \tau)$ (Section 7).

On Solana, verification uses the native `alt_bn128` syscalls for addition, scalar multiplication, pairing and point (de)compression.

### 2.4 Merkle trees

A *UTXO tree* of depth $d$ is an append-only binary Merkle tree over $\mathbb{F}_r$ with node function $\mathsf{H}(\cdot,\cdot)$; membership of a leaf $\mathsf{cm}$ under root $\mathsf{rt}$ is shown by an authentication path of $d$ siblings. An *indexed Merkle tree* stores a sorted linked list of values, each leaf holding $(v,\ \mathsf{next})$; non-membership of $x$ is shown by a *low leaf* $(v, \mathsf{next})$ with $v < x < \mathsf{next}$ and its authentication path. Indexed trees support constant-size non-membership proofs and are used for nullifier sets.

## 3 The substrate: Solana Privacy Rings

SPR is a UTXO-based confidential transfer protocol executed by an on-chain shielded pool program (SPP). We summarise the parts B2 depends on.

### 3.1 UTXOs and commitments

An SPR UTXO is a tuple

$$u = (\mathsf{domain},\ \mathsf{owner},\ \mathsf{asset},\ \mathsf{amount},\ \mathsf{blinding},\ \mathsf{utxo_data},\ \mathsf{ring_data},\ \mathsf{ring_program_id})$$

where $\mathsf{owner}$ is an *owner hash*

$$\mathsf{owner_hash} = \mathsf{H}\big(\mathsf{hb}_{33}(\mathsf{0x53} \,\|\, \mathsf{pk}_{\mathrm{sig}}),\ \mathsf{nk}\big), \qquad \mathsf{nk} = \mathsf{H}(\mathsf{nullifier_secret}),$$

binding a Solana signing identity $\mathsf{pk}_{\mathrm{sig}}$ (an Ed25519 key or a program-derived address) to a nullifier public key. The UTXO commitment is

$$\mathsf{cm}(u) = \mathsf{H}\big(\mathsf{domain},\ \mathsf{tree_id},\ \mathsf{asset},\ \mathsf{amount},\ \mathsf{data_hash},\ \mathsf{ring_hash},\ \mathsf{H}(\mathsf{owner}, \mathsf{blinding})\big),$$

with $\mathsf{asset} = \mathsf{hb}_{32}(\mathsf{mint})$ (the all-zero address for SOL) and $\mathsf{data_hash}$ an application-defined commitment to $\mathsf{utxo_data}$ that SPR includes but does not interpret. Its nullifier is

$$\mathsf{nf}(u) = \mathsf{H}\big(\mathsf{cm}(u),\ \mathsf{blinding},\ \mathsf{nullifier_secret}\big).$$

Because $\mathsf{nullifier_secret}$ is bound into $\mathsf{cm}(u)$ through $\mathsf{owner_hash}$ and is never published, $\mathsf{nf}(u)$ is unlinkable to $\mathsf{cm}(u)$ for anyone who does not hold the secret.

### 3.2 State

Commitments are appended to a **depth-32 Poseidon UTXO tree** (capacity $2^{32} = 4{,}294{,}967{,}296$ leaves) whose program retains a history of the last 500 roots, so proofs remain valid while concurrent transactions advance the tree. Nullifiers are inserted into a **height-40 indexed nullifier tree** (capacity $2^{40}$) fed by an on-chain queue and updated in batches by a separately proven batch-append circuit. A nullifier can be inserted exactly once.

### 3.3 The SPP proof

A `transact` instruction carries a Groth16 proof $\pi_{\text{pool}}$ whose single public input is

$$\mathsf{pih} = \mathsf{HashChain4}\big(\mathsf{RightHashChain4}(\mathsf{nf}_{\text{in}}),\ \mathsf{RightHashChain4}(\mathsf{cm}_{\text{out}}),\ \mathsf{tree_slot_chain},\ \mathsf{output_tree_id},\ \mathsf{private_tx_hash},\ \mathsf{external_data_hash},\ \ldots\big).$$

Among other conditions, the circuit enforces for every real input: owner-hash binding, Merkle inclusion of $\mathsf{cm}$ under a committed root, correct derivation of $\mathsf{nf}$, and non-inclusion of $\mathsf{nf}$ in the nullifier tree; for every output: a well-formed commitment with a derived blinding; and per asset, *balance conservation*

$$\textstyle\sum_{\text{in}} \mathsf{amount} + \mathsf{deposit} = \sum_{\text{out}} \mathsf{amount} + \mathsf{withdrawal}.$$

$\mathsf{external_data_hash} = \mathsf{Sha256BE}(\cdot)$ binds the proof to the exact instruction data and the settlement accounts, so a proof cannot be replayed against a different instruction, recipient or funding account.

The *private transaction hash*

$$\mathsf{private_tx_hash} = \mathsf{H}\big(\mathsf{in_chain},\ \mathsf{out_chain},\ \mathsf{addr_chain},\ \mathsf{private_tx_blinding}\big)$$

commits to all real input and output commitments under a secret blinding. SPR's ZK-program interface requires any application proof that accompanies an SPP proof to commit to the same $\mathsf{private_tx_hash}$; this is the hook B2 uses (Section 5.1).

The pool proof is serialised in 192 bytes: $A$ and $C$ as compressed $\mathbb{G}_1$ points (32 bytes each) and $B$ as an uncompressed $\mathbb{G}_2$ point (128 bytes).

### 3.4 Encryption

Output notes are encrypted with an HPKE-style scheme: a per-transaction X25519 key, derived deterministically from the transaction's first nullifier, performs ECDH with the recipient's viewing key; HKDF-SHA256 with a per-transaction 16-byte salt and the slot index yields an AES-256-CTR key and nonce. Integrity follows from recomputing $\mathsf{cm}$ from the decrypted plaintext and comparing it with the proof-verified on-chain commitment.

### 3.5 What the default ring reveals

The default ring is *confidential, not anonymous*: amounts and assets are hidden, but every output is tagged with its owner's public key so that wallets can find their UTXOs by a single indexed lookup. Senders and recipients of default-ring transfers are therefore public. This is the property B2 removes.

## 4 Architecture

### 4.1 Overview

B2 consists of four components (Figure 1):

**Vault program**
An on-chain Solana program that owns, at the SPR layer, all value deposited through B2; verifies vault proofs; enforces deposit limits; and authorises SPP state transitions by signing as its program-derived address via `invoke_signed`.

**Desktop wallet**
Holds the user's keys, maintains a local mirror of the SPR trees, generates both proofs and builds transactions.

**Fee relay**
A stateless submission service that pays the network fee for private transfers and unshields, so that no account of the user signs them.

**SPR**
The shielded pool program, its trees and its verifier, used unmodified.

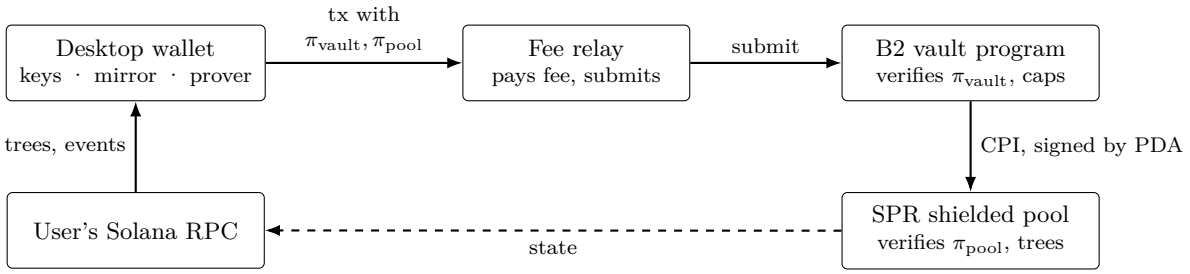


**Figure 1:** Components of a private B2 transfer. The wallet never signs a private transfer; the relay signs only as fee payer; the vault program signs only as the SPR owner.

### 4.2 The program-owned vault

In the default ring every UTXO's owner identity is public. B2 assigns every B2-held SPR UTXO to one and the same signing identity: a program-derived address $P$ of the vault program. At the SPR layer, therefore, every B2 transfer is a confidential transfer from $P$ to $P$, and the owner tag that the default ring publishes carries no information about the user.

Per-user ownership must then be enforced elsewhere: since $P$ is controlled by the vault program, any party able to make the program sign could otherwise spend any B2 UTXO. The vault program signs as $P$ only after verifying a *vault proof* $\pi_{\text{vault}}$ which establishes, in zero knowledge, that the caller knows the B2-level secret that authorises the specific inputs being spent.

### 4.3 Keys and addresses

A B2 wallet derives all of its key material from a single recovery phrase: the secrets that authorise spends and derive nullifiers, and a viewing key used to detect and decrypt incoming notes. A B2 address (`b2:...`) encodes the public counterparts a sender needs to pay its owner. Addresses are reusable by design: every payment is bound to a fresh one-time key derived from the recipient's address, so the notes of different payments to the same address share no public value. Recipients detect incoming payments locally, with their viewing key, while synchronising.

### 4.4 Operations

B2 exposes four user operations. Their public footprint is summarised in Table 1.

**Shield**
Moves SOL from the user's public Solana account into the vault. The deposit is necessarily public (source account, amount, asset, time); the resulting note and its future spends are not linked to it on chain.

**Transfer**
Spends one or two of the sender's notes and creates notes for the recipient and for change. Submitted by the relay.

**Unshield**
Spends notes and releases value to an arbitrary Solana address. Destination and amount are public, the originating notes are not. Submitted by the relay, or directly by the user (self-rescue path, Section 8).

**Receive**
Passive: the wallet detects and decrypts notes addressed to the user while synchronising.

## 5 Circuits

B2 defines six vault circuits, each compiled to R1CS over $\mathbb{F}_r$ and proven with Groth16. Table 2 lists their sizes as compiled for the mainnet deployment.

| Operation | Public on chain | Hidden |
|---|---|---|
| Shield | source account, amount, asset, time | resulting note, its owner and later spends |
| Transfer | that a B2 transaction occurred; relay as fee payer; vault PDA as SPR owner | sender, recipient, amount, asset, input notes |
| Unshield | destination account, amount, asset, time | sender, input notes, remaining balance |
| Receive | nothing (off-chain) | everything |

**Table 1:** Public footprint per operation.

| Circuit | Constraints | Proving key | Verifying key | Proving time[†] |
|---|---|---|---|---|
| `shield` | 7,920 | 1,528,542 B | 296 B | 910 ms |
| `transfer1` | 7,945 | 1,532,382 B | 296 B | 856 ms |
| `transfer2` | 11,054 | 2,285,406 B | 296 B | 1,397 ms |
| `release` | 8,420 | 1,870,206 B | 296 B | 969 ms |
| `deposit` | 15,190 | 2,895,166 B | 296 B | 1,364 ms |
| `unshield` | 9,144 | 1,986,366 B | 296 B | 1,160 ms |

**Table 2:** Vault circuits. [†]Median of five runs, release build, proof generation only (after key load, before self-verification), Apple M5, 10 cores, 32 GB.

### 5.1 Binding the two proofs

A private B2 transaction carries $(\pi_{\text{vault}}, \pi_{\text{pool}})$. Both proofs are verified in the same Solana transaction, which executes atomically: either both state transitions apply or neither does. SPR's ZK-program interface requires the application proof to commit to the SPP transaction through `private_tx_hash`, which in turn commits to every real input and output commitment of the pool transition. Consequently a vault proof generated for one pool transition cannot be combined with a pool proof for another.

## 6 On-chain verification and accounting

### 6.1 Proof formats

The vault proof is serialised fully compressed in 128 bytes: $A$ (32-byte compressed $\mathbb{G}_1$), $B$ (64-byte compressed $\mathbb{G}_2$), $C$ (32-byte compressed $\mathbb{G}_1$). Together with the 192-byte pool proof, a private transaction carries **320 bytes of proof**. Each vault verifying key occupies 296 bytes.

### 6.2 Compute

Measured on four mainnet transactions:

| Transaction | Compute units | Share of the 1.4 M per-transaction limit |
|---|---|---|
| Private transfer (a) | 243,112 | 17.4 % |
| Private transfer (b) | 241,374 | 17.2 % |
| Unshield (a) | 251,192 | 17.9 % |
| Unshield (b) | 252,935 | 18.1 % |

**Table 3:** Compute consumed by complete B2 transactions, both proofs included. Each paid the 5,000-lamport base fee.

### 6.3 Deposit limits and conservative accounting

The mainnet build of the vault program enforces, on chain:

- a per-deposit limit of 1 SOL;
- a limit of 25 SOL on the vault total $T$, stored in a dedicated state account that holds no authority;
- SOL as the only accepted asset.

Deposits increase $T$ and are rejected if they would exceed either limit. Releases are *always* permitted and decrease $T$ by the released amount. Releases to one-time keys inside the protocol do not decrease $T$. Hence $T$ is an upper bound on the value actually held: the limit can only be reached early, never exceeded, and no limit can ever block a withdrawal.

## 7 Trusted setup

The vault circuits' proving and verifying keys were generated by the B2 team with a purpose-built key-generation tool using radix-2 evaluation domains for the QAP, and are recorded with their hashes in the repository. Mainnet keys are distinct from all keys used on test networks; as a control, a proof made with test keys was submitted against the mainnet program and correctly rejected.

As in every Groth16 deployment, soundness relies on the setup randomness having been destroyed after key generation. Privacy does not depend on it: Groth16 proofs are perfectly zero-knowledge for every well-formed CRS, and subversion zero-knowledge is obtained by checking CRS well-formedness [3]. The current keys come from a single-party setup; they will be replaced by keys from a public multi-party ceremony, which is sound as long as at least one participant is honest.

The pool circuits' keys belong to SPR and are maintained by its developers; B2 uses the key set of the deployed SPP.

## 8 Network layer: the fee relay

A Solana transaction must be signed and paid for by a fee payer, whose address is public. If users paid their own fees, every private transfer would reveal a user account. B2 therefore routes private transfers and unshields through a relay that signs *only* as fee payer.

The relay is a stateless serverless function with a small persistent ledger. Before co-signing it

- accepts only transactions addressed to the B2 vault program;
- simulates the transaction and rejects it if its cost to the fee payer exceeds a bound (0.01 SOL);
- enforces a daily spending budget (1 SOL) and a rate limit (20 submissions per hour per address);

and returns either the signature or a typed error (`BAD_TX`, `NOT_ALLOWED`, `TOO_EXPENSIVE`, `RATE_LIMITED`, `BUDGET_EXHAUSTED`, `SEND_FAILED`). It keeps no request logs and sets no cross-origin permissions.

The relay learns the transaction, which it can also read on chain, and the client's IP address and request time. It cannot learn keys, balances or counterparties, and it cannot alter a transaction without invalidating the proofs, which are bound to the instruction data through `external_data_hash`. If the relay is unavailable or censors a user, the wallet can unshield directly, paying the fee from the user's public account; this trades the hidden-fee-payer property for liveness.

## 9 Client

### 9.1 Local proving

Both proofs are generated on the user's machine; spending keys and witnesses never leave it. Pool-proof generation, including the local tree synchronisation it requires, took 142–236 ms in the same runs as Table 2. End-to-end proving for a one-input transfer is therefore about one second on Apple M5, and below two seconds for every circuit.

### 9.2 Indexer-free synchronisation

The wallet connects to a Solana RPC endpoint chosen by the user and checks the cluster's genesis hash before use. It reconstructs the UTXO and nullifier trees locally from on-chain data rather than querying an indexer, so no indexer learns which notes belong to the user, and no indexer can withhold or forge state. Because the wallet retrieves the full commitment set rather than querying for specific notes, the RPC provider learns

nothing about which notes are the user's. The reconstruction is incremental from a persisted tree checkpoint:

| Synchronisation | Time |
|---|---|
| Nothing new since last start | 0.40 s |
| 103 new transactions | 19.3 s |
| First start, from the release snapshot | 28.8 s |
| First start before checkpointing | 21 min |

**Table 4:** Wallet synchronisation against Solana mainnet.

## 10 Security

### 10.1 Adversary model

We consider an adversary that observes the entire chain, controls arbitrary Solana accounts and RPC traffic it originates, may operate or compromise the fee relay, and may collude with any set of B2 users other than the honest parties under consideration. We do not consider compromise of the user's device, and we treat network-level anonymity (IP addresses, timing of requests) as out of scope for the cryptographic layer (Section 11).

### 10.2 Assumptions

**Assumption 1** (Groth16)**.** Groth16 over BN254 is knowledge-sound for the vault and pool circuits, given that the setup randomness of the respective key sets is not known to the adversary.

**Assumption 2** (Poseidon)**.** Poseidon over $\mathbb{F}_r$, with the parameters used by SPR, is collision-resistant and behaves as a random oracle for commitments and nullifiers.

**Assumption 3** (Discrete logarithms)**.** The computational Diffie–Hellman problem is hard in X25519, and discrete logarithms are hard in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of BN254. BN254 is the curve behind Ethereum's pairing precompiles and Solana's `alt_bn128` syscalls; after the exTNFS improvements [4] its security level is estimated at about 100 bits.

### 10.3 Properties

**Proposition 1** (Balance)**.** *No adversary can withdraw more value from the vault than was deposited into it.*

*Argument.* Every state transition carries a valid pool proof, which enforces per-asset balance conservation and nullifier uniqueness at the SPR layer; nullifier non-membership and one-time insertion prevent double spending. The vault program signs as $P$ only for transitions accompanied by a valid vault proof bound to the same `private_tx_hash`. Under Assumptions 1–2 the adversary can therefore spend only notes whose B2-level secret it knows, and only once.

**Proposition 2** (Sender anonymity)**.** *A private transfer reveals no information about which note was spent beyond membership in the set of all B2 notes present in the tree at the referenced root.*

*Argument.* Inputs are proven by Merkle membership under a public root without revealing the leaf; published nullifiers are pseudorandom functions of secrets never revealed; both proofs are zero-knowledge; the SPR-level owner of every B2 note is the same identity $P$; and the fee payer is the relay.

**Proposition 3** (Recipient anonymity and unlinkability)**.** *Notes created for a recipient are indistinguishable from notes created for any other B2 user, and two payments to the same B2 address are unlinkable on chain.*

*Argument.* Output commitments are hiding through their random blindings and the per-payment one-time keys described in Section 4.3; ciphertexts are produced under fresh per-transaction keys. In a test of twelve payments to one B2 address, analysis of the resulting on-chain data found no link between any two of them.

**Proposition 4** (Confidentiality)**.** *Amounts and assets of private transfers are hidden.*

*Argument.* Amount and asset enter only Poseidon commitments and ciphertexts; the public amount slots of the pool proof are zero for internal transfers.

### 10.4 Anonymity set

The anonymity set of a spend is the set of B2 notes in the tree at the referenced root: every note owned by $P$ is a valid candidate for every spend, and the set grows with each new note. As in every shielded system, shield and unshield events are public by construction; users maximise their cover by shielding ahead of time rather than immediately before spending, and by avoiding distinctive amounts at these boundaries.

## 11 Trust model and roadmap

### 11.1 Trust model

**Setup**
Vault-proof soundness relies on the destruction of the setup randomness (Section 7); privacy does not.

**Program governance**
The vault program is upgradeable through a dedicated upgrade key, held separately from every operational key, so that improvements can ship without migrating user funds.

**Relay**
The relay pays fees and submits transactions. It cannot spend, redirect or alter them, and users can always bypass it (Section 8). It sees the network origin of requests; for network-level anonymity, users can connect through Tor or a VPN.

**Deposit limits**
On-chain limits of 1 SOL per deposit and 25 SOL in total bound the value at risk while usage grows; withdrawals are never restricted (Section 6).

### 11.2 Roadmap

- Public multi-party setup ceremony for the vault circuits.
- Independent security audit of the vault program and circuits.
- Higher deposit limits.
- Additional assets.
- Additional platforms beyond macOS on Apple Silicon.

## A Deployment

| Item | Value |
|---|---|
| Network | Solana mainnet |
| Vault program | `E8bEh76PeW959PQTa27YkzZhhkPcgBJwegY6QW43zmz6` |
| State account | `B2jzvP9AXmgZENoWgAmEZpQKSN3C8AMjN2McqA6xNo8h` |
| Upgrade authority | `BYGYY8X1s8wqCXkZbWRuP9Spj4XERqbpwxAqLrcnycZB` |
| Deployment slot | 474,579,040 |
| Program binary | 119,040 bytes; on-chain bytes verified identical to the tested build |
| SPR crates | `spr-program`, `spr-interface`, `spr-tree` 0.10.0 |

## B Parameters

## References


[1] J. Groth. *On the Size of Pairing-Based Non-interactive Arguments.* EUROCRYPT 2016.

[2] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, M. Schofnegger. *Poseidon: A New Hash Function for Zero-Knowledge Proof Systems.* USENIX Security 2021.

[3] G. Fuchsbauer. *Subversion-Zero-Knowledge SNARKs.* PKC 2018.

| Parameter | Value |
|---|---|
| Curve | BN254 ($r \approx 2^{253.6}$) |
| Proof system | Groth16 |
| Hash | Poseidon over $\mathbb{F}_r$; SHA-256 for external data |
| UTXO tree | depth 32, $2^{32}$ leaves, 500-root history |
| Nullifier tree | indexed, height 40 |
| Proofs per private transaction | 2 (vault + pool) |
| Proof bytes per private transaction | 128 + 192 = 320 |
| Vault verifying key | 296 bytes |
| Note encryption (SPR layer) | X25519, HKDF-SHA256, AES-256-CTR |
| Per-deposit limit / total limit | 1 SOL / 25 SOL |


[4] T. Kim, R. Barbulescu. *Extended Tower Number Field Sieve: A New Complexity for the Medium Prime Case.* CRYPTO 2016.

[5] D. Hopwood, S. Bowe, T. Hornby, N. Wilcox. *Zcash Protocol Specification.*

[6] A. Langley, M. Hamburg, S. Turner. *Elliptic Curves for Security.* RFC 7748, 2016.

[7] H. Krawczyk, P. Eronen. *HMAC-based Extract-and-Expand Key Derivation Function (HKDF).* RFC 5869, 2010.