The Alberta Buck - Notes (DRAFT v0.2)

Private Bearer Instruments on an Identity-Aware Demurrage Currency

Perry Kundert

2026-04-18

For forty years cryptographers have tried to build digital cash and failed. Chaum's DigiCash (1990s) had the mathematics but went bankrupt for want of merchant adoption. Bitcoin gave us digital scarcity but no native bearer-paper form – Casascius coins and BIP-38 paper wallets are aftermarket wrappers, not first-class instruments. Central bank digital currencies threaten the opposite: digital surveillance with no bearer privacy at all.

The Alberta Buck identity, reserve, and demurrage machinery – combined with one new primitive, a global commitment pool – finally permits a native digital-private bearer instrument. This document specifies that primitive.

A **BUCK Note** is an on-chain commitment whose opening authorises a private mint to a registered identity. The same commitment can live only on chain (a software-only transfer between two of a user's own private accounts), travel between parties as a digital message, or be printed on paper as a QR-encoded slip. All three surfaces share the same cryptography and the same on-chain footprint.

A single Merkle tree of note commitments and a single nullifier set replace every parallel bearer-instrument architecture the Alberta Buck series has previously explored. Two orthogonal axes parameterise a note – *depositor Identity binding* (present: only the named recipient may deposit; absent: any registered holder of the opening may) and *issuer visibility* (public: the issuer's Identity `M` is in the clear on the note; private: it is encrypted to the recipient). Crossing these yields three realisable flavors (the private-bearer cell is ruled out by BUCK's mutual-decryptability invariant):

- **A2 – Addressed Cheque, Private Issuer.** The issuer re-randomizes the recipient's registered ElGamal credential into the note payload; the A-spend predicate requires the depositor's *current* registered credential to decrypt to the same identity point. A note Alice issues from a private account to a named recipient (including herself at another of her own accounts) with

no on-chain edge visible to third parties. The canonical *wormhole* is the degenerate self-issued subcase.

- **A1 – Addressed Cheque, Public Issuer.** Same addressed binding as A2, plus a Schnorr signature by the issuer over the commitment with their plaintext `M_issuer`. A cheque whose maker is publicly attributable and whose recipient is named but not visible to third parties – payroll, audit-friendly disbursement, B2C payment.
- **B1 – Bearer Cheque, Public Issuer.** Plaintext issuer `M`; no recipient binding; first registered holder to publish the nullifier ρ= redeems. A cashier's cheque: maker known, bearer not, cash-on-presentation.

Demurrage needs no new machinery. The Note Pool *is* simply the BUCK ERC-20 contract's own balance – a regular BUCK account accumulating BUCK-age like any other account. Mint: the user transfers v BUCKs to the contract (a standard ERC-20 transfer; sender pays their own outgoing demurrage fee), incrementing both the contract's balance B and a scalar F (sum of face values of live notes). Redemption is an *age-preserving transfer* unique to the Notes contract's single egress path: the recipient's nominal balance rises by the *full* face value v, and their BUCK-age rises by $v \cdot A/B$, where A and B are the pool account's accumulated BUCK-age and balance in the existing ERC-20 ledger. That is: the pool's *current* age-per-BUCK follows the redeemed value to the recipient, who then bears the pool's share of demurrage on their subsequent outgoing transfers via the standard BUCK accounting. This is exactly the cost a user would have paid had they held v BUCKs personally through the pool's current average residence period. Donations to the contract dilute A/B (donations add balance with no age), directly lowering the age-share absorbed by all subsequent redemptions. Conservation is automatic: balance and age are each exactly transferred, nothing is created or destroyed; the Jubilee continues to grow autonomously at $\kappa \cdot \text{totalSupply}$. No face discount, no division in the delivered amount, no F in the formula – only a single new primitive (age-preserving transfer), localized to one contract function.

A BUCK Note can be materialised as a physical paper slip with a QR-encoded opening, a hologram seal or scratch-off panel over the secret, and an issuer mark – indistinguishable in use from a $20 bill while being cryptographically impossible to counterfeit. Demurrage solves physical cash's oldest problem: a lost note's value decays at the same rate as an on-chain balance, with the difference accruing to the Jubilee rather than as seigniorage to the issuer. Issuance is on chain and identity-bound; circulation is private; deposit is auditable. This is stronger compliance than physical cash provides today, and stronger privacy than any CBDC has ever proposed.

The consequence most relevant to users: the *addressed* flavors (A1, A2) are cryptographically safe for long-term holding, because the spend predicate binds the recipient's Identity `M` and no mint-time knowledge enables a non-recipient to spend. The *bearer* flavor (B1) is a civil-trust instrument: the issuer necessarily knows ρ and could race the holder to deposit, so B1 cheques are meant to be cashed promptly and trusted only to the extent the issuer's legal standing and operational discipline warrant. This is an intrinsic cryptographic property, not a policy choice. Notes are designed to defeat *bulk on-chain traceability* – KYC-provider subpoenas, chain-analytics fingerprinting, omnibus surveillance – while preserving *bilateral* Identity disclosure: the two named parties to any specific note always learn each other's `M`, and either is compellable on warrant. (PDF, Text)

Contents

1 The Cash Paradox

Physical cash is simultaneously irreplaceable and impossible to digitize.

It is irreplaceable because it does things no digital instrument has ever matched. A $20 bill works without a network, a battery, a bank, an ID check, or a counterparty to your transaction. It clears instantly, finally, and without fees. It is the single most successful monetary instrument in human history – so successful that roughly half the value of US currency in circulation is held overseas as a global store of value, in defiance of every theoretical reason it shouldn't be.[1]

It is impossible to digitize because every attempt has reproduced the surveillance properties of bank money instead of the privacy properties of cash. Modern card networks log every transaction. Bank wire transfers require both parties to be identified. Stablecoins record every transfer publicly on chain. Central bank digital currencies, where they exist or are being designed (China's e-CNY, Europe's digital euro proposals), are explicitly programmable surveillance tools – the central bank can see, restrict, expire, or geofence any holder's balance at will.[2]

The closest existing analogues to digital cash are aftermarket bearer wrappers around cryptocurrencies: Casascius physical Bitcoin coins (2011-2013), OpenDime USB sticks, BIP-38 paper wallets, and the various scratch-card "crypto gift cards" sold at convenience stores. These work, technically – a Casascius coin is a genuine bearer instrument – but they are workarounds, not designs. The underlying token (Bitcoin) does not natively understand bearer-paper form. Loss of the physical coin is permanent loss of the value. Counterfeiting requires faith in the manufacturer's hologram, not in the cryptography.[3]

What if the bearer-paper form were a *first-class* instrument of the system? Designed in from the start, with the same cryptographic and economic properties as on-chain balances, integrated with the identity and reserve mechanics that already exist?

That is what BUCK Notes is.

1.1 Why Forty Years of Trying

David Chaum invented digital cash in 1982 with the blind-signature e-cash scheme.[4] By 1989 he had founded DigiCash to commercialize it. By 1998 DigiCash was bankrupt – not because the cryptography failed (it worked), but because no merchant wanted to be the first to accept it and no consumer wanted to be the first to hold it. The two-sided market never bootstrapped.

Stefan Brands extended Chaum's work in 1993 with offline divisible coins that revealed the spender's identity if (and only if) they double-spent.[5] Camenisch, Hohenberger, and Lysyanskaya improved on this in 2005 with compact divisible e-cash.[6] None of these schemes ever shipped in

[1] Federal Reserve. "Currency in Circulation" (2024). Approximately half of US currency in circulation by value is held outside the United States, providing a global store-of-value function despite zero yield.

[2] Bank for International Settlements. "Rise of the Central Bank Digital Currencies" (2020). Surveys CBDC designs across major central banks; notes that nearly all incorporate explicit issuer-side surveillance and programmability features (spending limits, geofencing, expiration, freezing) that physical cash does not have.

[3] Caldwell, M. "Casascius Physical Bitcoins" (2011-2013). Physical brass and silver coins containing Bitcoin private keys under tamper-evident holographic seals. Production halted in 2013 after FinCEN classified the issuer as a money transmitter requiring federal licensing. Surviving "unpeeled" Casascius coins now trade at substantial premiums above their Bitcoin face value as collectibles.

[4] Chaum, D. "Blind Signatures for Untraceable Payments" (1982). *Advances in Cryptology - Crypto '82*, pp. 199-203. The foundational paper introducing blind signatures and digital cash. Chaum's DigiCash company commercialized the design from 1989-1998 before bankruptcy.

[5] Brands, S. "Untraceable Off-line Cash in Wallets with Observers" (1993). *Advances in Cryptology - Crypto '93*, pp. 302-318. Offline divisible e-cash with double-spend identity revelation – a deterrent rather than prevention model.

[6] Camenisch, J., Hohenberger, S., Lysyanskaya, A. "Compact E-Cash" (2005). *Advances in Cryptology - Eurocrypt*

production at scale.

Bitcoin, when it arrived in 2009, was deliberately *not* digital cash in the Chaumian sense. It was digital scarcity with public ledger transparency – closer to digital gold than digital cash. The bearer-paper market that grew up around it (Casascius, OpenDime, paper wallets) was an afterthought, not a design goal.

Zerocash (2014)[7] and its production deployment as Zcash gave us the first scalable on-chain commitment-and-nullifier UTXO model – the cryptographic primitive that BUCK Notes use. Zcash showed that you can have private on-chain balances; it did not extend that to printable physical bearer instruments. Aztec[8] extended the model with predicate-encrypted notes for private smart contract calls, but again, no paper.

Tornado Cash[9] reduced the primitive to its simplest form (fixed-denomination commitments in, anonymous withdrawals out) and was OFAC-sanctioned in 2022. Privacy Pools[10] extended Tornado with per-spend association-set proofs, restoring compliance paths without abandoning the privacy primitive.

The reason all of this never produced BUCK Notes – a printable, demurrage-aware, identity-aware private bearer instrument – is that it required four pieces to come together simultaneously, and they did not until now:

1. A **commitment-and-nullifier UTXO model** on a smart contract platform (Zerocash gave us this in 2014; Aztec deployed it on Ethereum in 2018-2020).

2. A **cryptographic identity layer** that lets some notes be targeted to a specific recipient and others be self-wormholed between a user's own unlinkable accounts (the BUCK identity system, with PS signatures, ElGamal credentials, and Chaum-Pedersen re-encryption – see Identity).

3. A **demurrage mechanism** that solves the lost-cash problem and gives bearer notes a finite lifespan (BUCK's demurrage and Jubilee are unique among monetary systems).

4. A **jurisdictional reserve** that backs the value, so the bearer note represents a real claim on real assets, not a speculative crypto-token whose price might collapse between print and deposit (Alberta Buck's commodity-basket-stabilized reserve and the BuckK PID controller).

Alberta Buck has all four. No prior system had all four. That is why BUCK Notes is a *design*, not a workaround.

'05, pp. 302-321. Compact divisible e-cash where a single coin can be spent in pieces with O(1) communication and storage per spend.

[7]Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M. "Zerocash: Decentralized Anonymous Payments from Bitcoin" (2014). IEEE S&P 2014. The foundational design for on-chain commitment-and-nullifier UTXO privacy, deployed in production as Zcash since 2016 – the commitment-and-nullifier primitive used by BUCK Notes.

[8]Williamson, Z., et al. "Aztec Protocol" (2018-). Ethereum L2 rollup providing general-purpose private smart contracts using PLONK proofs and predicate-encrypted notes. The cryptographic ancestor of the `predicate` field in BUCK Note commitments.

[9]Pertsev, A., Semenov, R., Storm, R. "Tornado Cash Privacy Solution" (2019). Fixed-denomination commitment-and-nullifier mixer on Ethereum. Sanctioned by the US Office of Foreign Assets Control (OFAC) in August 2022 for alleged use in money laundering by North Korean state actors.

[10]Buterin, V., Schaefer, J., Tromer, E., et al. "Blockchain Privacy and Regulatory Compliance: Towards a Practical Equilibrium" (2023). Extends commitment-pool mixers with per-spend "association set" proofs, allowing depositors to demonstrate non-membership in incriminating subsets without revealing their specific deposit.

2 The Core Primitive: One Global Commitment Pool

The entire BUCK Notes design rests on a single on-chain primitive: a Merkle tree of commitments with a nullifier set, accompanied by an age-preserving transfer that delivers the opening's face value plus the pool's proportional BUCK-age to a recipient. Everything else – paper notes, addressed cheques, bearer cheques, self-wormholes – is a consequence of varying the *spend predicate* inside the commitment.

2.1 The Note Contract State

The BUCK contract gains exactly three new pieces of state:

State	Type	Purpose
`noteTree`	Merkle tree of commitments	Cryptographic anchor for every outstanding note
`nullifiers`	Set of used nullifiers	Double-spend prevention
`noteFaceSum`	F in $\mathbb{Z}$	Sum of face values of all outstanding notes

That is all. The pool's nominal BUCK balance B and accumulated BUCK-age A are not tracked as separate state: *the pool is just a BUCK account*, specifically the ERC-20 contract's own address, and B, A are exactly the balance and BUCK-age the ERC-20 ledger already maintains for it. No lazy integration, no shadow aggregates, no separate `poolT` timestamp. `noteFaceSum` is a pure audit scalar – the total face-value liability outstanding – and does not appear in the redemption formula itself.

BUCKs transferred to the contract address (via mint, donation, or accident) joins B through the normal ERC-20 path. BUCKs transferred out of the contract on redemption uses a *single new primitive*: an age-preserving transfer that moves both balance and proportional BUCK-age from the contract to the recipient. This is the only non-standard BUCK operation in the design.

2.2 The Note Commitment

A note commitment is a Poseidon hash of the fields the SNARK will later open:

$$cm \;=\; H_{\text{cm}}(\text{flavor},\; v,\; \rho,\; \text{id-payload},\; \text{predicate})$$

with:

- **flavor** $\in \{A1, A2, B1\}$: the Identity-binding flavor
- **v** $\in \mathbb{Z}_q$: face value
- ρ: per-note randomness (source of the nullifier)
- **id-payload**: Identity-binding data (see "Note Flavors" section)
- **predicate**: optional additional spend predicate (time locks, oracle conditions, multi-sig; not elaborated in detail here – a single bit `predicate == None` keeps the SNARK to the three base flavors. See "Open Questions" for bounded-programmability notes.)

No per-note timestamp or baked-in BUCK-age: demurrage is computed at redemption time from the *pool's* account state (B, A), not from the individual note's life history, so no per-note age fields need to ride in the commitment[11].

[11]The astute reader will recognize that this essentially *freezes* demurrage for BUCK Note amounts while they are in transit; this has beneficial effects for the transmission of large long-term BUCK balances, such as inheritances. The cost of this benefit is effectively amortized across all BUCK Note users; the value of Anonymity for short-term Note users subsidies long-term Note holders, who bear some additional risk that their Notes may be lost]

The commitment is inserted as a leaf of `noteTree` at mint time.

2.3 The Note Nullifier

The nullifier is a pseudo-random function of the note's secret witness material:

$$nf \;=\; \mathrm{PRF}_{\mathrm{key}}(\rho)$$

where `key` is the spend-side secret specific to the flavor: the depositor's sk_{dep} for A-spend (binding through CP-equality to the recipient's `M`), or ρ itself for B-spend (publicly derivable by any holder of the opening). The PRF is instantiated via Poseidon in the SNARK-friendly setting.

The nullifier is emitted on deposit; double-spend is rejected by set-membership check.

2.4 Lifecycle

```
mint (burn):
  # Standard ERC-20 transfer from caller to contract address.
  # Caller pays their own outgoing-transfer demurrage through the
  # normal BUCK accounting path.  Contract balance B gains v; its
  # accumulated BUCK-age A starts earning on the new v from here.
  transfer v BUCKs from caller to contract
  for each of {cm_1, ..., cm_N}:
     append cm_i to noteTree
  noteFaceSum += sum(v_i)

deposit (mint-out):
  verify SNARK (A-spend or B-spend, per flavor):
    - Merkle inclusion of some cm in noteTree
    - Knowledge of (v, rho, id_payload, predicate)
    - Flavor-specific spend predicate satisfied by witness
    - nullifier correctness
    - v is a public output of the SNARK
  require nf not in nullifiers
  nullifiers += nf
  # Read pool state from the ERC-20 ledger:
  (B, A) = (balanceOf(contract), buckAgeOf(contract))
  ageShare = v * A / B
  # Age-preserving transfer (the one new primitive):
  #   contract.balance -= v;   contract.age -= ageShare
  #   recipient.balance += v;  recipient.age += ageShare
  transferPreservingAge(contract, recipient, v, ageShare)
  noteFaceSum -= v
```


Full face value is delivered: the recipient's nominal balance goes up by exactly v. The pool's demurrage cost rides along as the BUCK-age $v \cdot A/B$ added to the recipient's account. The recipient will bear that demurrage on their subsequent outgoing transfers through the ordinary BUCK accounting path – the same path any holder travels, but on an age they acquired by accepting the note. No Jubilee payment is emitted from inside the deposit; the Jubilee grows autonomously at

$\kappa \cdot \text{totalSupply}$, which continues to include both B (before redemption) and v (after, as part of recipient's balance).

3 A Note is a Deferred Approve Handshake

The single most useful way to understand BUCK Notes is that *a note is an* `approve` *handshake split across time and mediated by the commitment pool*. Once this frame is in place, every structural choice below follows mechanically.

3.1 The Regular On-Chain Transfer

A standard BUCK transfer between two registered identities is a synchronous bilateral exchange:

1. Alice calls `approve(bob, v, E_alice_for_bob, CP_proof)`. She re-encrypts her credential under pk_{bob} and proves (Chaum-Pedersen) that the re-encryption carries the same M_{alice} as her registered credential.
2. Bob calls `approve(alice, 0, E_bob_for_alice, CP_proof)`. Same in the other direction.
3. `transfer(bob, v)` consummates the transfer once both halves are recorded.

All three steps may land in the same block, as long as they're available by `transfer(bob, v)`-time. Each party learns the other's M; every on-chain observer sees both addresses, the amount, and the timing. The transfer edge `(alice, bob, v, t)` is public and permanent.

3.2 The Note is the Same Handshake, Deferred

A Note reorganises the same three messages:

1. **Mint.** Alice performs *her half* of approve against a pool commitment rather than directly against Bob. Depending on whether the depositor is known at mint time, she either encrypts her credential under the depositor's public key (`E_alice_for_depositor`, identity-bound flavor) or publishes her credential in the clear with a bearer secret ρ (public-issuer bearer flavor). The commitment is appended to `noteTree`; Alice's outgoing BUCKs lands in the pool's balance B.
2. **Deposit.** The eventual depositor submits a SNARK-proved spend predicate that simultaneously:
   - opens the commitment and proves nullifier correctness,
   - completes *their half* of approve by attaching `E_depositor_for_issuer` (the mirror ElGamal ciphertext) as a deposit-event payload, and
   - authorises the age-preserving transfer of v and $v \cdot A/B$ out of the pool.
3. **Receipt.** The issuer later scans deposit events for the nullifier of the note she printed and reads `E_depositor_for_issuer` to learn who cashed it.

Both parties end up with the same cryptographic knowledge of each other that `approve` would have produced. The only difference from the observer's perspective is that mint and deposit are two independent transactions, and the commitment pool hides which mint corresponds to which deposit; the demurrage liability transferred with v is the Note pool average at that instant[11].

3.3 Mutual Decryptability: The BUCK Identity Invariant

Every BUCK transfer, note or otherwise, preserves a load-bearing Identity property: *both parties must end up able to decrypt each other's M*. In `approve`, this is achieved explicitly by the two re-encryption calls. In a Note, the same property must still hold:

- At *mint* time, the issuer has to publish, alongside the commitment, enough material that the eventual depositor can recover M_{issuer}.
- At *deposit* time, the depositor has to publish enough material that the issuer (reading chain events) can recover $M_{\text{depositor}}$.

This invariant is the single strongest constraint on note structure. It is what rules out *pure-bearer anonymous notes* (neither party learns the other's identity), and what forces the exactly-three-flavor taxonomy below.

3.4 What the On-Chain Observer Sees

At mint: a transfer of v BUCKs from Alice's address to the contract address, plus an `Appended(cm)` event. The commitment cm is a Poseidon hash; observers learn nothing about the opening, the depositor, the issuer flavor, or ρ.

At deposit: an age-preserving transfer of v BUCKs from the contract address to the depositor's address, plus a `Deposited(nf, E_depositor_for_issuer)` event. Observers learn who the depositor is (standard transfer visibility), but not which mint this corresponds to.

The two transactions are decoupled in time and unlinked in the chain state. Any mass-harvest analysis that tries to reconstruct the `(issuer, depositor, v, t)` edge must either (a) correlate timing across the whole pool anonymity set, (b) subpoena a party with direct knowledge, or (c) break the SNARK / Poseidon.

3.5 Why This Framing Matters

Almost every downstream design choice is a corollary of "deferred approve handshake":

- *Three note flavors* correspond to three valid ways to complete the two halves of the approve across time (next section).
- *Mutual decryptability* is the invariant that kills a fourth flavor (private issuer, unknown depositor).
- *Selective disclosure for compliance* is the natural consequence of each party still holding cryptographic receipts of its counterparty, exactly as in `approve`.
- *Integration with regular transfers* requires no new semantics at the contract boundary: a note deposit is an ERC-20 transfer from the contract address, and the identity-exchange payload in the event is structurally identical to the receipt fragment `approve` already stores.

4 The Surface of a Note: Chain, Message, Paper

A BUCK Note is, fundamentally, a tuple

```
(flavor, v, rho, id_payload, predicate, merkle_path)
```

that opens to a commitment $cm = H_{\text{cm}}(\text{flavor}, v, \rho, \text{id-payload}, \text{predicate})$ in the on-chain note tree. The tuple can live on any medium capable of storing ~200-500 bytes and being read back faithfully. Three surfaces are supported:

4.1 Chain-Only Notes

The opening is transmitted privately from issuer to depositor (for a self-wormhole A2, both are the same user operating from different accounts) through an out-of-band encrypted channel – typically the wallet's internal storage, or a direct peer-to-peer message between wallets. The note never leaves the issuer's control infrastructure. The canonical self-wormhole case: Alice mints an A2 note targeting her own `M` from a private account and deposits it into a second unlinkable account.

4.2 Message Notes

The opening travels as a digital message: an email, SMS, QR shown on a screen, signal-messenger attachment, NFC handoff. No physical artifact exists, but the recipient can store, forward, or print the opening at their convenience. A1 addressed cheques between parties with reliable digital communication occupy this surface.

4.3 Paper Notes

The opening is printed as a QR code on a slip of paper, a card, a steel plate, or a sticker. For addressed notes (A1, A2) – where the cryptography already prevents a photographing observer from spending – no physical seal is required. For bearer B1 cheques – where whoever reads ρ can spend – the QR must be protected by a tamper-evident seal (hologram, scratch-off panel, or laminated sleeve) so the holder can verify they were the first to expose the secret.

A printed note typically also carries:

- A visible denomination (1, 5, 10, 100, 1,000, 10,000 BUCK) for human convenience. The actual value is what the commitment encodes; the printed face is metadata that a wallet verifies before deposit.
- A human-readable issuer mark, for visual recognition. Cryptographic authenticity comes from the on-chain commitment, not the mark.
- Redundant QRs front and back with Reed-Solomon error correction, plus the secret in base32 digits for manual recovery.

Paper notes are the form most people will think of when they hear "BUCK Note." Physical paper is the surface of last resort: it works offline, without batteries, across borders, through power outages, and past the expiration of any given digital communication channel.

4.4 Counterfeiting is Cryptographically Impossible

A "fake" BUCK Note is any opening whose claimed commitment does not appear in the on-chain note tree. The Merkle inclusion proof inside the spend SNARK fails. No mint occurs. No second-guessing is required: there is no paper-inspection threshold to deceive and no "close enough" forgery that a harried cashier might accept.

This is a stronger anti-counterfeit guarantee than the Bank of Canada provides for physical $20 bills. Polymer banknotes can be counterfeited (the techniques are well-published; the question is only whether the counterfeit passes casual inspection). A BUCK Note cannot be counterfeited at all – the on-chain commitment registry is the authentic-bill list, and any printed note whose commitment is not in the tree will produce a `FAKE` wallet status on scan and nothing more.

4.5 Wallet-App Status Codes

A user-facing wallet that scans a BUCK Note (from chain storage, message, or paper QR) displays one of five statuses:

Display	Meaning
`VALID: 1,000 BUCK`	Commitment in tree, nullifier unspent, spend predicate satisfiable
`DEAD`	Commitment in tree, nullifier already spent
`FAKE`	Commitment not in tree
`EXPIRED`	Predicate includes an expiry that has passed
`NOT FOR YOU`	Addressed (A1/A2) note whose bound `M_rec` does not match the viewer's Identity

Casual users do not need to understand the underlying cryptography any more than they need to understand the security features of a $20 bill. They scan, see `VALID`, and accept the note as payment.

On Deposit, the wallet constructs the SNARK locally (or via a delegated prover service for thin clients), submits the transaction, and displays the result. SNARK construction for a single-note deposit is on the order of seconds on a modern smartphone – Aztec's Noir compiler and Zcash's Halo2 prover both run on mobile.

After successful deposit, the note's nullifier is consumed and the QR is permanently dead. A torn-in-half paper note is a *human* signal of finality, not a cryptographic one – but it's a useful ritual to prevent confused future holders from trying to redeposit a dead QR.

5 Note Flavors: Depositor Binding x Issuer Visibility

Every note is characterised by two orthogonal choices:

- **Depositor M-binding** – present or absent. When present, only identities encrypting the same target M may deposit (the spend predicate is a Chaum-Pedersen equality, *Theorem 6*). When absent, any registered identity may deposit (the spend predicate is knowledge of a bearer secret ρ plus a Schnorr PoK of the depositor's registered sk).
- **Issuer M visibility** – public or private. The issuer's Identity point M_{issuer} is either attached in the clear (plaintext plus Schnorr signature) or encrypted such that only the eventual depositor can recover it (ElGamal under the depositor's pk, re-randomised from the depositor's registered credential).

Crossing the two axes nominally produces four cells, but the *mutual decryptability* invariant of the previous section immediately rules out one of them:

Issuer / Depositor	Depositor M-bound (A)	Depositor M-unbound (B)
Public issuer (1)	**A1** – identified cheque to a specific registered recipient (payroll; audited disbursements; B2C with attribution)	**B1** – bearer cheque from a known issuer (pre-printed cashier's cheque; any registered identity deposits)
Private issuer (2)	**A2** – private cheque to a specific registered recipient (addressed cheque; self-wormhole is degenerate issuer == recipient)	**B2 – impossible**: a private issuer with no pre-known depositor has no pk under which to encrypt M_{issuer} for the eventual depositor to recover.

The three valid flavors – **A1, A2, B1** – together cover every legitimate pool-mediated approve handshake on an identity-aware currency. They are *not* three different cryptographic primitives; they are three compositions of the same two building blocks (Chaum-Pedersen equality and Schnorr PoK), distinguished by which half of the handshake is carried in the commitment.

5.1 Flavor A2: Addressed Cheque, Private Issuer

The canonical private-to-private transfer. Subsumes "self-wormhole" (the degenerate case issuer ≡ recipient, where Alice mints to her own registered M and later deposits from a different registered credential bound to the same M).

Mint. The issuer reads the recipient's registered credential $E_{\text{rec}}^{(\text{reg})} = (R_r, C_r)$ from the IdentityRegistry and re-randomises it with fresh r':

$$E_{\text{note}} \;=\; (R_r + r'G,\; C_r + r' \cdot pk_{\text{rec}}) \;=\; \big((r_r + r')G,\; M_{\text{rec}} + (r_r + r')pk_{\text{rec}}\big).$$

This is a valid ElGamal encryption of M_{rec} under pk_{rec} with randomness $r_r + r'$. The issuer never learns M_{rec} or r_r – only the on-chain ciphertext is used as a re-randomisation template. To complete her half of the deferred approve, the issuer also encrypts *her own* M_{issuer} under pk_{rec} as $E_{\text{iss-for-rec}}$ and packs it alongside the commitment.

$$\text{payload}^{A2} \;=\; \big(E_{\text{note}},\; E_{\text{iss-for-rec}}\big)$$

Scan. The recipient trial-decrypts each candidate note's E_{note} with her own sk_{rec} and checks whether the result equals her M_{rec} (which she knows). Exactly one ElGamal decrypt and one point-equality per candidate – the same scan cost as a stealth-address scheme.

Spend. The depositor (= recipient, or any credential sharing the same M) produces a SNARK proving:

1. Merkle inclusion of $cm = H_{\text{cm}}(\text{flavor},\ v,\ \rho,\ \text{payload}^{A2})$ in `noteTree`.
2. Nullifier $nf = H_{\text{nf}}(\text{flavor} \| sk_{\text{dep}} \| \rho)$.
3. Chaum-Pedersen equality (Theorem 6) between the depositor's registered $E_{\text{dep}}^{(\text{reg})}$ and the note's E_{note}: both decrypt to the same M.
4. `faceValue` $= v$.

Simultaneously, the depositor attaches `E_depositor_for_issuer` (an ElGamal re-encryption of their own M_{dep} under the issuer's pk_{issuer}) as a deposit-event payload, completing the approve handshake. The issuer's pk_{issuer} is not in the commitment (privacy!) but is carried inside the private witness and decrypted by the recipient from $E_{\text{iss-for-rec}}$.

Consequences. Only credentials encrypting M_{rec} can deposit (CP-soundness). The issuer cannot spend what she printed unless she herself is the recipient (issuer's credential encrypts $M_{\text{issuer}} \neq M_{\text{rec}}$; no valid CP proof). Key-loss recovery is automatic: Identity re-issuance gives the recipient a fresh sk'_{rec} bound to the same M_{rec}, and Theorem 6 still verifies.

5.2 Flavor A1: Addressed Cheque, Public Issuer

Identical to A2 except the issuer additionally publishes her own M_{issuer} in the clear, with a Schnorr signature binding it to cm:

$$\text{payload}^{A1} \ = \ \big(E_{\text{note}},\ m_{\text{issuer}},\ \sigma_{\text{issuer}}\big)$$

The depositor no longer needs `E_iss-for-rec` – she reads M_{issuer} directly. No privacy loss for the depositor: `E_depositor_for_issuer` still conceals who deposited from casual observers, because the deposit event is only linkable to a specific note by someone holding ρ or scanning the commitment pool exhaustively.

Public issuers are typically institutional (employers, charities, payroll processors). Their identity is already public; baking it into the note adds audit value without changing any cryptography.

Use cases. Payroll from a named employer to an identified employee, charitable disbursement to an identified grantee, attribution-preserving B2C payment where the customer wants a cryptographic record that a specific merchant paid them.

5.3 Flavor B1: Bearer Cheque, Public Issuer

The classic cashier's cheque. The issuer does not know the depositor at mint time; the note is bearer, and whoever shows up first with knowledge of ρ and a valid registered credential deposits.

$$\text{payload}^{B1} \ = \ \big(m_{\text{issuer}},\ \sigma_{\text{issuer}},\ pk_{\text{issuer}}\big)$$

No depositor-side ElGamal is pre-encrypted because no depositor is known. The depositor reads the issuer's identity in the clear; the issuer reads the depositor's `E_depositor_for_issuer` from the deposit event. Mutual decryptability holds – just entirely via the public-issuer side at mint, and via the deposit-event payload at deposit.

Spend. The depositor produces a SNARK proving:

1. Merkle inclusion of $cm = H_{\text{cm}}(\text{flavor},\ v,\ \rho,\ \text{payload}^{B1})$.
2. Nullifier $nf = H_{\text{nf}}(\text{flavor} \| \rho)$ (publicly derivable from ρ; this is the first-come-first-served property of bearer notes).

3. Signature-in-SNARK check: σ_{issuer} verifies against pk_{issuer} in IdentityRegistry.

4. Schnorr PoK of sk_{dep} for registered pk_{dep}, Fiat-Shamir bound to `msg.sender`, `chainid`, `nullifier`.

5. `faceValue` $= v$.

No CP-equality is required – the issuer is public, the depositor is any registered identity, and the approve handshake completes via plaintext M_{issuer} plus `E_depositor_for_issuer`.

Consequences. The issuer knows ρ and therefore *can* race the intended bearer to deposit. But any such deposit attaches the issuer's identity to the deposit event – so an issuer-turned-thief is *identified at the moment of theft*. This is a civil-trust problem (the issuer breaks a promise to the bearer), not a cryptographic vulnerability. Same trust model as a paper cashier's cheque: the bank can in principle redirect the draft, and the law stops them.

5.4 Comparison of the Three Flavors

Property	A2 Addressed, private issuer	A1 Addressed, public issuer	B1 Bearer, public issuer
Primary use	Private remittances; self-wormhole	Payroll; charity; B2C attribution	Cashier's cheques; pre-printed bea
Issuer known to observers	No (ElGamal; only depositor decrypts)	Yes (plaintext)	Yes (plaintext)
Depositor pre-known at mint	Yes (bakes in recipient's M)	Yes (bakes in recipient's M)	No (any registered identity)
Spend predicate	CP-equality (Theorem 6)	CP-equality (Theorem 6)	Schnorr PoK + signature-in-SNA
Issuer can spend own note	No (CP fails: different M)	No (CP fails: different M)	Yes, but identified on deposit
Long-term holding safe	Yes; key loss recoverable via Identity	Yes; key loss recoverable	Yes cryptographically; civil-trust
Mutual decryptability	$E_{\text{iss-for-rec}}$ at mint $+ E_{\text{dep-for-iss}}$ at deposit	Plaintext M_{iss} at mint $+ E_{\text{dep-for-iss}}$ at deposit	Plaintext M_{iss} at mint $+ E_{\text{dep-for-iss}}$ at deposit

The cryptographic taxonomy collapses to **two spend-circuit shapes**: an **A-spend** (CP-equality, covering A1 and A2) and a **B-spend** (Schnorr PoK + ρ, covering B1). The public-vs-private issuer distinction is commitment metadata, not a separate circuit.

A fourth flavor – bearer notes with no identity hook on either side – is intentionally absent; see "What This Architecture Rules Out" below.

6 The Interaction Matrix: Note Flavor x Depositor Account Mode

The Identity system (Identity) gives registered accounts two visibility modes:

- **Anonymous-valid.** KYC-verified but Identity encrypted. A counterparty learns `M` only by completing the bilateral `approve` handshake (or being targeted by a note whose payload pre-completes that handshake).
- **Public.** `M` is published in the clear. No handshake is needed to know who the account belongs to.

Permissionless (unregistered) accounts cannot appear on either side of a BUCK transfer – regular or Note – because the mutual-decryptability invariant requires a registered identity at every endpoint.

Crossing the three valid flavors with the two depositor account modes produces the six operational cells below. Rows are note flavors (A1 / A2 / B1); columns are the depositor's account visibility at deposit time. The *issuer's* account mode is fixed by the flavor itself: `A2` issuers are anonymous-valid (or the bearer-issuing contract acting as a pseudo-identity), `A1` and `B1` issuers are public.

Flavor	Anonymous-valid depositor	Public depositor
A2 (private cheque)	**Private-to-private addressed cheque** Issuer (anon-valid) targets a specific anon-valid recipient. No on-chain link between them; both learn the other's `M` at spend. Self-directed *wormhole* is the degenerate subcase (issuer = recipient). Use: remittance, inheritance, B2B payments.	**Private-to-public addressed cheque** Issuer (anon-valid) targets a public recipient. Recipient deposits to their public account; observers see only that a public identity received a pool note. Use: moving private wealth to a public persona; donating to a public charity without a traceable direct transfer.
A1 (identified cheque)	**Public-to-private identified cheque** Public issuer targets an anon-valid recipient. On-chain observers see the issuer's `M` plus a deposit from some anon-valid account; they cannot link the account to its real Identity. Use: payroll, B2C payments, audit-friendly disbursements from a public treasury.	**Public-to-public identified cheque** Both parties public; the note adds /timing indirection/ over a direct `transfer`: the cheque can be pre-signed and held. The issuer is already on-chain; the recipient may choose among their public accounts at deposit time. Use: pre-signed instruments, escrow cash-outs.
B1 (bearer cheque)	**Bearer-to-private** Public issuer prints an unaddressed cheque; first registered spender to publish ρ wins. Civil-trust instrument: theft is not cryptographically prevented, only bilaterally traceable. Use: cashier's cheques, physical coupons, prize vouchers, out-of-band bearer drafts.	**Bearer-to-public** Same, but depositor is public. Observers see a public identity cashed a B1 draft from issuer `M`. Use: redeeming a bearer instrument received out-of-band into a publicly attributable account.

Observations:

1. **Wormhole functionality** – the private same-identity round-trip is the degenerate subcase of A2 where issuer and depositor share the same `M`. It uses the same mint construction, the same spend circuit, and the same pool; no separate instrument is required.
2. **Cashier's cheques are B1** and are the only flavor where the recipient identity is *not* determined at mint time. All other flavors bind a specific recipient via the re-randomized `E_note` payload. This is also the only flavor exposed to issuer race-condition risk – whoever publishes the nullifier ρ first takes the value.

3. **Public-to-public A1 cells** are degenerate in information content: a direct `transfer` carries the same on-chain signal. Their utility is operational (pre-signing, mailing, delayed destination choice), not privacy-structural.

4. **Issuer and depositor account modes are chosen independently**, subject only to the flavor's constraint on the *issuer* side (A2 = private issuer; A1/B1 = public issuer). The depositor's mode affects only what the on-chain observer sees at deposit, never what the counterparties see of each other.

5. **The B2 cell (private issuer + bearer) does not exist.** A private issuer must re-randomize for a specific recipient's `pk_rec` so both parties can decrypt each other's `M`; there is no recipient to key against in a bearer construction. Attempting a "private bearer" instrument violates the BUCK mutual-decryptability invariant.

7 Integration with On-Chain Transfers

Notes are not a separate payment system. They are an alternative framing of a BUCK transfer in which the `approve` handshake is mediated asynchronously through the pool instead of being settled synchronously between two accounts. Everything that is true of a regular `transfer` – demurrage accounting, Jubilee conservation, sanctioned-account blocks, fee and gas mechanics – is also true of a Note mint or deposit, because both sides of a Note's lifecycle *are* regular ERC-20 transfers against the pool's own account.

7.1 The Pool is a Regular Account

The BUCK contract's own address is a registered account in the ordinary sense. BUCKs flowing in (mints, donations) is a standard incoming transfer; BUCKs flowing out (redemptions) is a standard age-preserving outgoing transfer. The only contract-local state not already tracked by the ERC-20 ledger is `noteTree` (the commitment Merkle tree), `nullifiers` (the spent-note set), and `noteFaceSum` (aggregate live face value – an audit convenience, not a liability figure).

7.2 Regular Transfer vs. A-Flavor Note

A regular `transfer(alice, bob, v)` followed (or preceded) by `approve(alice, bob)` has the following on-chain signature:

- Alice's account: balance debited by $v \cdot B_A/(B_A - \kappa A_A)$, demurrage settled at Alice.
- Bob's account: balance credited by v.
- Edge observable: Alice –> Bob, amount v, block t.

The A-flavor equivalent (A2 between two anon-valid accounts):

- Alice mints: balance debited by the same $v \cdot B_A/(B_A - \kappa A_A)$ (Alice pays her own outgoing-transfer demurrage, settling her age-per-BUCK at mint time). The pool's balance gains v, age-clean.
- Bob deposits at time $t + \Delta$: the pool transfers v face plus $v \cdot A_{\text{pool}}/B_{\text{pool}}$ proportional age to Bob. Bob now bears the pool's averaged age for the value he receives; his own account's age-per-BUCK adjusts upward in proportion.
- Edge observable: Alice –> pool at t; pool –> Bob at $t + \Delta$. The edge connecting them is erased.

Economic outcomes match a regular transfer up to the timing difference: Bob acquires age proportional to how long the pool held the value, which under pool-rate demurrage is bounded by the pool's current A/B regardless of how long Bob's specific commitment sat in the tree. This is the key property that makes indefinite holding of A-flavor Notes economically equivalent to holding BUCK – same demurrage rate, just with a different edge visible on chain.

7.3 Regular Transfer vs. B-Flavor Note

A B1 cashier's cheque replaces the bilateral `approve` with a public-issuer signature binding the note to a specific M_{issuer} and leaves the depositor unspecified:

- Issuer mints: `transfer` from issuer to pool. The issuer's m_{issuer} is bound into the commitment via the Schnorr signature.
- First registered holder to publish ρ deposits: pool –> depositor at some later time.
- The depositor and issuer end up mutually identified (per the BUCK invariant) through the note's commitment payload plus the on-chain deposit actor; no third party sees the edge.

A B1 Note is the natural digital analogue of a cashier's cheque drawn on a bank: the bank (public issuer) has pre-committed the funds; whoever presents the instrument redeems them.

7.4 When to Use Which

Scenario	Use	Why
Two parties already mutually approved	`transfer`	Simpler, one on-chain call, no SNARK cost
Payer wants no on-chain edge to recipient	A2	Breaks bulk-harvest trace; same demurrage
Public issuer paying a specific recipient	A1	Auditable issuance, recipient visibility hidden
Pre-signed instrument, recipient undecided	B1	Bearer; cash on presentation
Self-wormhole across own accounts	A2 (issuer = recipient)	Breaks the edge between own addresses
Payroll to many private employees	A1 per employee, or A2	A1 when attribution required, A2 otherwise
One-off payment under surveillance concern	A2	Cheapest way to erase the direct edge

7.5 Cost and UX Trade-Offs

Relative to a regular `transfer`, every Note costs additional gas for the SNARK verifier (mint or spend) and requires wallet software capable of producing the proof. For routine payments between counterparties who do not need to defeat chain analytics, a regular `transfer` is strictly cheaper and simpler. Notes are the correct tool when the privacy property – no on-chain edge between payer and recipient – is worth the additional gas and proving time, or when the instrument semantics (deferred recipient choice, pre-signed cheque, paper distribution) are themselves the point.

7.6 Composability

Notes compose transparently with the rest of BUCK:

- *Identity Fountain re-issuance* applies to Note mints and deposits identically to regular transfers; A-flavor key-loss recovery falls out automatically.
- *Sanctions* and *Jubilee* operate at the account level and reach the pool through its own account state, requiring no Notes-specific logic.
- *Demurrage* is accounted on each `B_X, A_X` pair including the pool's; pool-rate demurrage is a consequence of this accounting, not a parallel mechanism.
- *Wallet UX* can expose Notes as an alternative `send` mode in the same UI flow that calls `transfer`; the difference between "direct send" and "private send" becomes a single toggle.

8 Demurrage at the Pool Level

The central simplification of this design: **the Note Pool is just a regular BUCK account.** Specifically, it is the BUCK ERC-20 contract's own balance. BUCKs transferred to the contract's address join the pool; BUCKs transferred out (as note redemptions) leave the pool. The contract account accumulates BUCK-age like every other account and participates in the ordinary demurrage accounting with no special-case code. The only piece of additional state is a single scalar F, the sum of face values of live notes.

8.1 Reminder: How BUCK Demurrage Actually Works

Demurrage is *not transferred* on any per-account basis. It is an accounting entry maintained on each account's state:

- Every block, the Jubilee account's balance grows by $\kappa \cdot \text{totalSupply} \cdot \Delta t$. This is a single global operation; the sum of all other accounts' nominal balances is held constant.
- Each account X has a running BUCK-age A_X that grows at rate B_X (its nominal balance) per unit time.
- When account X transfers v BUCK out, its spendable fraction is $(B_X - \kappa A_X)/B_X$; the outgoing transfer costs the account $v \cdot B_X/(B_X - \kappa A_X)$ of its nominal balance, and the corresponding $\kappa A_X \cdot v/(B_X - \kappa A_X)$ demurrage "owed" is simply extinguished from (B_X, A_X) against the Jubilee's autonomous growth.

System-wide invariant (preserved by every transfer):

$$\sum_{\text{all accounts } X} B_X \;+\; B_{\text{Jubilee}} \;=\; \text{constant}.$$

The Jubilee balance reflects the aggregate accrual $\kappa \int_0^t \text{totalSupply}(s)\, ds$; individual accounts' κA_X quantities track *their share* of that total.

8.2 The Pool Account's Demurrage

Treat the BUCK contract address as an ordinary account. Let:

- B = the contract's ERC-20 balance (= sum of all BUCK transferred in via mints and donations, minus all BUCK transferred out via redemptions, modulo the standard demurrage adjustment on outgoing transfers).
- A = the contract's accumulated BUCK-age, as maintained by the ERC-20 ledger.

B and A are already on chain; the Notes design queries them, it does not compute them. The pool's *effective spendable value* at any moment is

$$E \;=\; B - \kappa \cdot A.$$

This is the amount the contract would be able to transfer out if it emptied itself in a single call – exactly the same definition that applies to any other account.

8.3 The Age-Preserving Redemption

When a note of face v is redeemed, the contract performs an *age-preserving transfer* of v BUCK from itself to the recipient:

Side	Balance change	BUCK-age change
Pool	$-v$	$-v \cdot A/B$
Recipient	$+v$	$+v \cdot A/B$

Full face value is delivered; the pool's BUCK-age *proportional share* – one average age-per-BUCK times face v – moves along with the balance to the recipient. The recipient's account absorbs that age just as if they had personally held v BUCK through the pool's residence period.

The pool's *own* age-per-BUCK ratio A/B is preserved across the redemption: \(A_{new}/B_{new} = (A - vA/B) / (B

- v) = A(B - v) / (B(B - v)) = A/B \). The ratio drifts only

through time (age accrues at rate B) and through mints (fresh BUCK arrives age-free, diluting the ratio).

Contrast with a standard BUCK transfer, where the sender bears the demurrage on their side (balance reduced by $v \cdot B/(B - \kappa A)$) and the recipient receives a clean v with no age. The age-preserving variant simply *follows* the balance with its proportional age, deferring the demurrage reckoning to the recipient's next outgoing transfer. This is an economically natural choice for a Note recipient: they have accepted the note in lieu of a direct BUCK holding, and in return accept the demurrage cost that goes with having held that BUCK.

8.4 Donations Integrate Smoothly

Any BUCK transferred to the contract address *without* minting a corresponding note increases B without adding BUCK-age (the sender's age stays with the sender per standard ERC-20 semantics). The pool's ratio A/B drops; subsequent redemptions carry a *smaller* age-share per face unit. Over a sequence of redemptions the donation is consumed as a demurrage subsidy for live noteholders.

Four origin cases, all handled identically:

1. **Accidental transfers.** A user fat-fingers the contract as a recipient. Their BUCK is not burned; it dilutes A/B, and future redeemers bear marginally less age.
2. **Philanthropic donations.** A user deliberately sends BUCK to the contract to subsidise noteholders. The subsidy spreads across all future redemptions until A/B drifts back up.
3. **Recovered collateral.** Mechanism-level refunds (e.g., an issuer's collateral unlock) sent to the contract address.
4. **Rounding / dust.** Integer-arithmetic residuals. Self-healing.

After all notes are eventually redeemed, any remaining donation principal sits at $F = 0, B = D_{\text{residual}}, A = 0$ – unclaimable by redemption, but available to subsidise a fresh round of mints if the pool is used again.

8.5 Conservation is Automatic, and Tight

The age-preserving transfer conserves both quantities *exactly*:

- $\Delta B_{\text{pool}} + \Delta B_{\text{recipient}} = 0$.
- $\Delta A_{\text{pool}} + \Delta A_{\text{recipient}} = 0$.

Total BUCK in circulation is unchanged by redemption; total accumulated BUCK-age is unchanged by redemption. The Jubilee continues to grow autonomously at rate $\kappa \cdot \text{totalSupply}$ per block; this total is unaffected by whether the v BUCK sits in the pool or in the recipient's account. And whether the demurrage on that age is eventually paid out by the pool's (hypothetical) transfer or by the recipient's real outgoing transfer, the same $\kappa \cdot v \cdot A/B$ amount of spendable value will be reconciled against the Jubilee.

No pool-specific conservation theorem is needed beyond the system-wide $\sum_X B_X + B_{\text{Jubilee}} = \text{const}$ (automatic) and the local observation that the age-preserving transfer is two-sided balanced (evident from its definition).

8.6 Numeric Example

Suppose the pool holds $B = 10\,000\,000$ BUCK with $A = 5\,000\,000$ BUCK-years (average age 0.5 yr), and $F = 10\,000\,000$ BUCK of live notes outstanding.

Alice redeems a note of face $v = 1{,}000$:

- Alice's balance rises by 1,000 BUCK.
- Alice's BUCK-age rises by $1{,}000 \cdot 5{,}000{,}000/10{,}000{,}000 = 500$ BUCK-years. In nominal units: her 1,000 incoming BUCK has an effective *age* of 0.5 yr attached to it.
- When Alice next spends 1,000 BUCK from this account, the outgoing transfer reconciles its share of her total BUCK-age against the Jubilee: roughly $0.02 \cdot 500 = 10$ BUCK worth of demurrage is paid out of her balance (the exact amount depends on her other holdings' age). Net effect: she receives 1,000, pays ~10 when she spends them, nets 990. This matches the previous-model result where she received 990 at redemption – same economic total, different accounting.

Now suppose an anonymous donor sends 500,000 BUCK to the contract address. B rises to 10,500,000; the donor's BUCK-age stays with them. The pool's A/B drops from 0.5 to $5{,}000{,}000/10{,}500{,}000 \approx 0.476$. Alice's next redemption absorbs a smaller age-share per face – a direct subsidy from the donor.

No Jubilee payment was emitted by the pool during any of this. The Jubilee's growth is unchanged: it keeps accruing $\kappa \cdot \text{totalSupply}$ per block regardless of who holds which BUCK.

8.7 Who Bears the Demurrage, Really

The answer is now sharply localized: **the Note recipient bears the pool's accumulated share of demurrage on the face value they received, and they pay it when they next spend from that account.** The recipient has accepted the note, and with it has accepted the demurrage cost of the BUCK sitting in the pool while the note was outstanding.

This is the "price" the recipient pays for accepting BUCK Notes instead of a direct BUCK transfer: they inherit the note's pool-residence demurrage, prorated across all live notes at the

moment of redemption. The trade-off is explicit – and visible in the resulting (B, A) state on the recipient's account.

Noteholders (while holding) bear no demurrage: they hold a cryptographic note, not a BUCK balance, and the BUCK they implicitly own sits in the pool where its age is borne collectively. The issuer (at mint time) bears their own outgoing-transfer demurrage to deposit into the pool, as for any ERC-20 transfer. Third parties bear nothing. The Jubilee keeps receiving the BUCK-wide demurrage stream $\kappa \cdot \text{totalSupply}$ regardless of how BUCK sloshes between accounts.

9 Demurrage Solves the Lost-Cash Problem

This section is about a property that is genuinely novel, with implications well beyond the BUCK system.

9.1 The Hidden Inflation of Lost Cash

Every monetary system that uses physical bearer instruments suffers a constant slow loss to *unrecoverable currency*. Banknotes are lost in fires, dropped down storm drains, forgotten in the pockets of jeans that get washed and shredded, buried in landfills, sealed in walls, hidden by the elderly and forgotten by their heirs.

The Federal Reserve estimates that 5-10% of US paper currency is permanently unrecoverable at any given moment.[12] At a US M0 of approximately $2.3 trillion, this represents *$115-230 billion in permanently extracted value* – seigniorage to the Federal Reserve, since the lost notes never get redeemed but the backing assets remain on the Fed's books.

This is not theft, but it is not neutral either. Lost cash is a regressive transfer: the people most likely to lose cash are the people who use cash most (the elderly, the unbanked, low-income workers paid in cash). The beneficiary is whoever issued the cash, generally the central bank, ultimately the government.

9.2 What BUCK Demurrage Does For Lost Notes

A BUCK Note that is lost in a fire, dropped down a storm drain, or hidden in a wall by a paranoid grandparent is *not* permanently extracted from the system. The face value v sits in the pool as part of B, earning BUCK-age A at the standard rate. No redemption ever happens.

Two effects balance:

1. The BUCK backing the lost note stays in the pool indefinitely, accruing age. It does not vanish.

2. Other live-note redeemers, through the age-preserving transfer, absorb an age-share $v \cdot A/B$ at each deposit. Because the lost note's share weighs on A without a corresponding redemption to drain it, the lost note's implicit demurrage is paid installment-style by subsequent redeemers.

Over decades, the pool's age-per-BUCK A/B drifts upward as lost notes accumulate unredeemed. The autonomous Jubilee growth at rate $\kappa \cdot \text{totalSupply}$ continues regardless. Net effect: *the 5-10% of currency that physical cash systems lose forever, BUCK Notes returns to the commons within a generation*, via the steady demurrage pressure that the lost face value applies to live redeemers.

This has cascading consequences:

1. **Money supply self-correction.** No system-wide accounting tail of "missing" currency that the central authority must track and write off. Lost notes write themselves off through the pool-age mechanism.

[12] Federal Reserve Bank of San Francisco. "Diary of Consumer Payment Choice" (2018). Estimates 5-10% of US paper currency by value is permanently unrecoverable. At a US M0 of $2.3 trillion, this represents $115-230 billion in permanently extracted seigniorage.

2. **Anti-hoarding incentive matches on-chain balances.** A BUCK held in your wallet or in your pocket as paper decays at the same rate (via the age-share that will eventually be absorbed on redemption). There is no advantage to converting on-chain BUCK to paper to evade demurrage; the demurrage clock runs the same way in both forms.

3. **Inheritance is automatic.** A grandparent who hides 100,000 BUCK in paper notes and forgets to tell anyone does not extract 100,000 BUCK from the economy on their death. The value flows back through the Jubilee to BUCK issuers – broadly, back into the same economy where the grandparent originally earned it.

4. **No central bank profit from currency loss.** The seigniorage that accrues to central banks today from unrecoverable cash does not exist in BUCK Notes. Lost value does not benefit the issuer; it benefits the system as a whole.

5. **Counterfeit recovery is automatic.** Even if a note printed years ago were somehow forged (which is cryptographically impossible, but humour the counterfactual), the value at stake is bounded by the demurrage decay. An attacker forging old notes would be forging paper whose backing BUCK has already been largely absorbed by live redemptions.

This is, I believe, the first monetary system in history where lost cash returns to the commons by automatic mechanism rather than by accounting fiat. Physical cash cannot have this property, because physical cash has no clock. It emerges naturally from BUCK's existing demurrage mechanism applied to the note-pool primitive.

9.3 The Symmetry With On-Chain Balances

A second elegance worth pointing out: the effective demurrage rate on a BUCK Note – as absorbed by the recipient at deposit – is *the same* as the demurrage rate on an on-chain BUCK balance that lived in the pool for the same duration. This is by design, and it matters.

If paper BUCK Notes demurraged faster than on-chain BUCK, holders would prefer on-chain (and paper would be discouraged, defeating the design's purpose). If on-chain BUCK demurraged faster than paper notes, holders would race to print all their on-chain balances as physical notes (and the chain would lose its primary use as a settlement layer).

The matching demurrage rate makes the choice between on-chain and paper purely about *use case fit*: on-chain for routine commerce, automated payments, large transactions; paper for offline payments, gifts, savings under the mattress, disaster preparedness, and so on. Neither form is privileged by the demurrage mechanism.

10 Long-Term Holding Safety: Addressed vs. Bearer

Pool-rate demurrage makes every flavor equally cheap to *hold*. What distinguishes them for long-term storage is whether the spend predicate binds a specific recipient Identity `M_rec` or accepts any registered depositor. Addressed notes (A1, A2) are cryptographically safe for indefinite holding; bearer notes (B1) are civil-trust instruments and should be cashed promptly.

10.1 What the Issuer Knows

At mint time the issuer holds the full commitment preimage $(v, \rho, \text{id-payload}, \text{predicate})$ including any randomness r' it generated to re-randomize the recipient's encrypted Identity. The question is whether that knowledge, by itself or combined with any later-acquired secret, lets anyone other than the intended recipient spend the note.

10.2 A-Flavors (Addressed): Cryptographically Safe

Both A1 and A2 bind a specific recipient at mint time. The note carries the re-randomized ElGamal ciphertext $E_{\text{note}} = (R_r + r' \cdot G,\ C_r + r' \cdot pk_{\text{rec}})$ of M_{rec} under the recipient's pubkey. The A-spend predicate is a Chaum-Pedersen proof of equality (Theorem 6) that the depositor's *registered* credential decrypts to the same M_{rec} that E_{note} encrypts.

The consequence is that *no amount of mint-time knowledge helps a non-recipient spend*:

- The issuer knows r' and can re-derive E_{note}, but cannot produce a CP-equality proof without a registered credential encrypting M_{rec}.
- An attacker who copies ρ cannot spend either; the pool contract binds ρ to the recipient's side of the equality.

A1 and A2 differ only in whether M_{issuer} is on-chain in the clear (A1) or encrypted for the recipient (A2). Neither affects who can spend.

10.2.1 Key-Loss Recovery via Identity Re-Issuance

BUCK Identity allows a user to re-issue their Identity scalar: the same registered M is re-attested with a fresh signing keypair (sk', pk'). Because A-flavor notes bind `M_rec` (not pk_{rec} in any structural way – the randomness r' attaches to the *old* pk_{rec} used at mint, but the CP-equality check runs against *any* currently-valid registered credential for the same M), a recipient who loses their mint-time private key recovers full spend ability as soon as they complete Identity re-issuance.

The implementation consequence: the A-spend circuit verifies equality against the depositor's *current* registered credential for M_{rec}, looked up from the IdentityRegistry at deposit time, not against the specific pk_{rec} snapshot the issuer used at mint.

Addressed notes are therefore the natural instrument for long-term holding: estate planning, cold storage, and the self-directed *wormhole* (A2 with issuer = recipient). Cryptographic loss of `M` itself – which would require the Identity system's own recovery mechanism – is the only failure mode, and it is shared with every other BUCK holding.

10.3 B1 (Bearer): Civil-Trust, Not Cryptographic

The B-spend predicate is a Schnorr PoK of knowledge of ρ plus any valid registered depositor credential. There is no recipient binding: whoever publishes the nullifier first wins the deposit. The

issuer, having minted the note, knows ρ; so does anyone who later observes ρ – including anyone the issuer accidentally leaks it to (database row, printed backup, compromised signing device) and anyone who photographs an unsealed physical cheque.

This is not a cryptographic failure; it is the *point* of a bearer instrument. The B1 flavor replaces cryptographic recipient binding with civil-law recourse:

- The issuer's Identity M_{issuer} is public on the note. An issuer who prints a cheque and then races to deposit it is visibly on chain as the double-spender. The registered Identity is subpoena-able and the issuer cannot repudiate having printed the note.
- A holder who discovers their B1 has been cashed by someone else has a named counterparty to pursue in civil or criminal court. The chain proves who deposited; the note's commitment and signature prove who issued it.

The practical rule: **treat B1 like a cashier's cheque**. Cash within hours to days. Use only issuers whose operational discipline and legal standing you are willing to rely on. Do not use B1 for long-term savings, inheritance, or cold storage; addressed A1 or A2 instruments serve those needs with cryptographic rather than institutional trust.

10.4 Summary: Which Flavors Can Be Saved?

Flavor	Can a non-recipient spend?	Long-term holding safe?	Comparable instrument
A1	No (CP-equality on M)	Yes	Registered bond / named cheque
A2	No (CP-equality on M)	Yes	Named bond, self-wormhole
B1	Yes (anyone who knows ρ)	No	Cashier's cheque / bearer bond

The consequence: *addressed (A) notes are cryptographically safe to hold; bearer (B1) notes rely on issuer operational discipline and civil recourse.* Placing an A1 or A2 cheque in cold storage for a decade trusts only the cryptography and the recipient's Identity recovery capability. Placing a B1 in the same vault trusts the issuer not to retain ρ for ten years.

Mapping to use cases:

- **Long-term savings, cold storage, self-wormhole:** A2 (you are the recipient).
- **Estate, inheritance, named gifts:** A1 or A2 – A1 when the recipient and heirs want public attribution of the issuer; A2 when discretion is preferred.
- **Payroll, B2C disbursements, audit-visible transfers:** A1, cashed at the recipient's pace.
- **Bearer instruments – cashier's cheques, physical coupons, prize vouchers:** B1, cashed promptly.

11 Physical Security Model

The cryptography only protects what the cryptography touches. The physical paper itself needs a security model. This section sketches one.

11.1 Layered Defense

A printed BUCK Note has three concentric layers of security:

Layer	What it protects	Mechanism
Inner: secret	The cryptographic ability to spend	QR code containing ρ
Middle: physical seal	Bearer (B1) publication-exclusivity	Hologram, scratch-off, laminate sleeve
Outer: identity binding	Addressed (A1/A2) targeting (only named spender)	CP-equality on the recipient's re-randomized ElGamal credent

The *inner* layer is the cryptographic core: as long as ρ (and the flavor-specific material packed into `id-payload`) is unknown, the note cannot be spent. This is unconditional protection against any threat that does not include observing those secrets.

The *middle* layer matters only for bearer B1 notes, where the spend predicate is "any registered holder of ρ." An unsealed B1 QR photographed by any observer allows that observer to race to spend. Sealing the QR under a tamper-evident covering means the note can be displayed (passed hand-to-hand, stored in a wallet, mailed in an envelope) without exposure. The seal must visibly damage on opening so that recipients can verify they were the first to expose the secret. (This does not rescue B1 from the issuer-race risk – the issuer who printed the note could still have retained ρ before sealing – which is why B1 remains unsafe for long-term holding. The seal addresses post-printing observation, not issuer trust.)

The *outer* layer makes addressed A1/A2 notes safe to *publicly* distribute. An addressed note can be photographed with no security implication: the photographer cannot spend it without also holding a registered credential encrypting the bound `M_rec`. The targeting binding is cryptographic, not physical. For addressed flavors, no physical seal is required.

11.2 Recommended Manufacturing Profile

A practical BUCK Note specification, suitable for industrial-scale printing:

- **Substrate.** Cotton-blend banknote paper or polymer (Bank of Canada-style). Steel or titanium plates for high-denomination "permanent" notes.
- **Printing.** Standard inkjet or laser printing of QR codes is sufficient for cryptographic correctness. For visual anti-confusion (so people can tell a legitimate note from a printout), use UV-reactive ink, microprinting, or watermarks – the same techniques used for traditional banknotes. These do not add cryptographic security; they add *reassurance*.
- **Seal.** Peelable hologram covering the secret QR (bearer B1 only), or scratch-off panel covering a separate code (also B1), or no seal at all (addressed A1/A2, since the cryptography handles exclusivity).
- **Redundancy.** Print the QR twice (front and back) with Reed-Solomon error correction. A note that loses one QR to wear or damage can still be deposited via the second. For higher-value notes, also print the secret as base32 digits in human-readable form.

- **Issuer mark.** A non-cryptographic logo identifying the issuer (the Alberta Buck reserve, a credit union, a commercial issuer). This is for visual recognition only; the cryptographic authenticity check is the on-chain commitment.
- **Denomination indication.** Visible face value for human convenience. The actual value is what the on-chain commitment encodes; the printed face value is metadata. A wallet app that scans an apparent 1,000 BUCK note and reads "actual value: 100 BUCK" from the commitment warns the user before deposit.

12 Use Cases

The same primitive serves a spectrum of use cases that today require entirely separate financial instruments. Each maps to one of the three realisable flavors (A1, A2, B1) according to whether recipient binding and public issuer attribution are required.

12.1 Walking-Around Money

For digital "send at the moment of payment," an A2 note targeted to the payee's registered `M` is ideal: the recipient receives a message-note that only they can deposit, and the issuer's Identity is exchanged bilaterally through the commitment payload without any on-chain edge. For truly paper bearer cash – a note that anyone who holds the paper can spend – a B1 cheque with hologram seal fills the role, issued by a trusted public party (credit union, commercial issuer) who publicly warrants erasure of ρ after printing.

Most walking-around use cases – paying for a coffee, tipping, splitting a bill with a friend – are better served by A2 message notes exchanged at the moment of payment. The paper B1 form is the right choice when one party has no digital wallet at hand or when strict bearer-on-paper semantics are required. If you are going somewhere that would be best to avoid taking your phone, or just because you want to retain complete anonymity, such a note might be appropriate.

Notes always yield the current pool demurrage rate, so neither encourage nor discourage rapid circulation rather than hoarding. The wallet-app status codes give immediate VALID / DEAD / FAKE feedback on scan. These public issuer / any depositor Notes always bear some non-zero risk of theft or an untrustworthy issuer who retained a copy of the note, so should be deposited promptly.

12.2 Cross-Border Remittances

A paper BUCK Note is just paper. It crosses borders in pockets, envelopes, and parcels at zero marginal cost. An A2 note targeted to the recipient's registered `M` makes the transfer useless to anyone except the intended recipient, so loss in transit is not a financial risk.

This compares favorably to the existing remittance industry, which extracts roughly 6.5% in fees globally on $700+ billion in annual transfers.[13] A paper-note remittance has a fixed cost: print

- mail. The mail might cost $1; print costs fractions of a cent;

deposit gas costs a few dollars at most. At any reasonable remittance value, the fee compression is enormous.

The recipient does not need a smartphone or internet at the moment of receipt – they need them only at the moment they want to convert the note to on-chain BUCK or to local fiat. In countries with intermittent connectivity, this is a major usability improvement over digital remittance services.

12.3 Tip Jars and Charitable Donations

A sheet of small-denomination B1 bearer notes, perforated for tear-off, can be left next to a tip jar or a donation box. Patrons take notes (denominated at, say, 2 BUCK each), scan them with their phone, and the value flows to the jar's registered identity.

[13] World Bank. "Remittance Prices Worldwide" (Q4 2024). Average global cost of remitting $200 was approximately 6.5% in 2024. Total global remittance flows estimated at $702 billion in 2024.

This works without any intermediary, payment processor fee, or opening of an account. The patron pays in paper; the recipient receives in on-chain BUCK. B1 is appropriate here because the turnaround is fast: the jar's operator empties the day's receipts to their main account nightly.

12.4 Cold Storage and Disaster Preparedness

An A2 self-issued note (issuer = recipient, both the same registered `M`), stored in a fireproof safe, is a more durable form of cold storage than a 12-word seed phrase. The note is *self-evidently* an instrument of value – a customs officer, a relative, or your future self immediately recognizes a paper labeled "1,000 BUCK Note" as something worth keeping safe. A seed phrase, by contrast, is opaque to anyone who has not been trained to recognize it.

For disaster preparedness: a small folder of BUCK Notes in a fireproof safe survives infrastructure failures that would render exchange accounts and even hardware wallets unusable. As long as the chain is reachable somewhere in the world, the notes can be deposited. In a true grid-down emergency, the notes circulate hand-to-hand at face value (with the matching pool-rate demurrage clock running uniformly, so prices remain stable in BUCK terms even during a multi-week outage).

A2 (issuer = recipient) is the correct choice for self-directed cold storage; A2 or A1 targeted to a designated heir's `M` is the correct choice for estate-directed cold storage. Identity re-issuance makes these robust against loss of the mint-time signing key.

12.5 Estate Planning

A2 notes targeted to your heir's registered `M`, stored in a safe-deposit box: zero probate process, zero tax-event delay, zero court intervention. The heir scans on your death, deposits, the BUCK is theirs. A1 is the same instrument with the issuer publicly attributable – appropriate where the heir or executors need the origin on-chain for tax or legal reasons.

For larger estates, additional conditional-spend predicates (time locks, death-certificate oracle attestation) can be composed into the commitment's `predicate` field. The SNARK does not care what the predicate is as long as it is well-defined at mint time.

12.6 Gift Cards Without Issuer Risk

A traditional gift card carries the risk that the issuing merchant goes bankrupt between purchase and redemption (cumulative consumer losses to bankrupt gift-card issuers run into the hundreds of millions annually in North America[14]). An addressed A1 or A2 BUCK Note carries no issuer risk: the value is on chain in the BUCK reserve, not on the merchant's books.

Specialized merchant gift cards become unnecessary – replaced by BUCK Notes of appropriate denomination, optionally bundled with a paper merchant receipt indicating the intended use.

12.7 Tax-Compliant Compensation

An employer paying a contractor in BUCK Notes faces the same tax-reporting obligations as paying in cash: payment is income and must be reported. The on-chain issuance trail makes this easier to comply with than paper cash, not harder – the employer's mint transaction is an auditable record of the payment, even though the contractor's deposit is private.

For employees: employers can issue A1 notes targeted to each employee's registered Identity, automating payroll while preserving employee privacy from external observers.

[14] Consumer Federation of America. "Gift Cards: A Costly Convenience" (2023). Estimates of cumulative consumer losses to bankrupt gift-card issuers in the United States range from $200M-$1B annually depending on methodology.

12.8 Underbanked and Cash-Economy Inclusion

This is, perhaps, the most important use case.

The Bank of Canada estimates 1-2% of Canadians are unbanked, with rates an order of magnitude higher in rural and Indigenous communities.[15] Globally the World Bank estimates 1.4 billion adults are unbanked.[16]

For these populations, paper cash is not a privacy preference; it is the only viable financial instrument. A digital-only currency (CBDC, stablecoin, traditional e-payment) excludes them by definition. BUCK Notes includes them: a person without a bank account, without a smartphone, without literacy in the local language, can hold and transfer BUCK Notes by physical handover, exactly as they hold and transfer paper currency today.

When and if they choose to interact with the digital economy, they deposit a note and the value moves on chain. Until then, they participate in the BUCK economy on exactly the terms they participate in the cash economy. The issuer of the paper – a credit union, a reserve-backed issuer like ATB, or a local community organisation – takes on the role that the central bank currently plays for physical fiat, while the recipient retains the anonymity of a cash user.

[15]Bank of Canada. "2022 Methods-of-Payment Survey" (2023). Approximately 1-2% of Canadians lack bank accounts; rates substantially higher in rural and Indigenous communities.

[16]World Bank. "The Global Findex Database 2021" (2022). An estimated 1.4 billion adults globally remain unbanked, with disproportionate concentrations in low-income countries, women, rural populations, and the elderly.

13 Privacy: Defeating Mass Harvest, Preserving Bilateral Disclosure

BUCK Notes do not make transactions anonymous. They do not hide who paid whom from the parties themselves: the mutual-decryptability invariant means the issuer and the eventual depositor always learn each other's `M`. They also do not hide a specific transaction from a properly targeted investigation: a warrant served on either named party to a Note compels disclosure of the counterparty's `M` and the surrounding payments.

What Notes do is *break the on-chain bulk-harvest channel*. Aggregate transfer graphs, KYC-provider subpoenas scoped across thousands of clients, and commercial chain-analytics fingerprinting all operate by correlating on-chain addresses into behavioural profiles without a specific target. Notes replace the on-chain address-to-address edge with a mint/deposit pair separated in time and dissociated by the pool, leaving the bulk observer only the deposits made *directly to or from that observer's own clients*. Everything else becomes a commitment landing in the pool and a nullifier leaving it.

13.1 What the On-Chain Observer Loses

Consider an analytics firm that purchases KYC-linked address data from a large wallet provider. Today, such a firm can reconstruct much of a BUCK user's economic life: regular payroll deposits, recurring vendor relationships, counterparties' other clients, timing patterns. Under Notes, the on-chain trace of a typical legitimate transaction is:

1. Payer's account transfers BUCK to the Notes contract (mint). The observer sees only that *some amount* entered the pool.
2. Some time later – minutes to months – a recipient deposits a note. The observer sees only that *some amount* left the pool to *some* registered account.

There is no observable edge between payer and recipient. The payer's account activity shows only "sent N BUCK to pool." The recipient's account activity shows only "received M BUCK from pool." Correlating the two requires either (i) complicity of one of the two parties, or (ii) subpoenaed disclosure by either party.

13.2 What Bilateral Parties Keep

The BUCK Identity invariant – that both endpoints of any value transfer can decrypt each other's `M` – is preserved. The payer learns the recipient's `M` when they read the recipient's IdentityRegistry entry to construct E_{note}. The recipient learns the payer's `M` when they verify the note (from m_{issuer} for A1/B1, from the paired $E_{\text{issuer-for-rec}}$ ciphertext for A2).

Neither party sees the other's wider on-chain activity from the Note alone. The payer sees only the specific `M` of the recipient, not the recipient's other counterparties or balances. The recipient sees the same narrow slice of the payer's life.

13.3 The Mass-Harvest Threat Model

The threat Notes defeat is *undifferentiated bulk observation*:

- A KYC-provider subpoena scoped to "all transactions of clients X, Y, Z for the last five years" yields only direct pool-to-client and client-to-pool transfers. It cannot reconstruct who X paid, or who paid Y, or whether X and Y ever transacted.
- A chain-analytics firm selling behavioural fingerprints cannot build the edge graph at all; there are no edges to cluster.
- An insurer, employer, or landlord who gains read access to a counterparty's on-chain address learns that address's pool-inflow and pool-outflow amounts, not its counterparties.

The threat Notes *do not* defeat is *targeted investigation with proper authorization*:

- A warrant served on party X to a specific Note compels X to reveal the counterparty's `M` and the associated context.
- A criminal probe that identifies a "mule" account depositing many Notes can compel that mule to disclose each note's issuer `M` (the income source) and each outgoing note's recipient `M` (the onward payment), walking the graph one cooperating party at a time.
- Sanctioned or frozen Identities are blocked at the contract level regardless of Note use.

This is the posture of a bank cheque or personal cheque, not a $20 bill: the parties to any specific instrument are identified to each other and compellable on individual warrant, but no omnibus order reaches across the system.

13.4 The Regulatory Audit Path

A regulator investigating illicit flows has a clean single-target path:

1. Identify a suspect account (by KYC on a wallet provider, a prior investigation, or a direct complaint).
2. Compel the suspect's wallet, via warrant, to disclose the plaintext `M` of every counterparty the suspect has transacted with via Notes, together with the note metadata (issuer / recipient, amount, time). This is entirely derivable from material the suspect already possesses: their own registry keys and the payloads they decrypted at scan or mint time.
3. Serve the next warrant on the resulting named counterparty. Repeat.

The audit is depth-first on named parties, not breadth-first across the chain. The *origination* of any specific note remains traceable through the on-chain mint record plus the issuer's own records; a bearer $20 bill has no analogous trail.

13.5 What Compliance Looks Like

A reasonable regulatory regime around BUCK Notes:

- *Issuers* must be identity-registered and may face reporting obligations on large issuances (analogous to current cash-withdrawal reporting).
- *Depositors* face the reporting obligations that attach to cash deposits at a bank today.

- *Holders* who pass notes hand-to-hand have no reporting obligation, the same way there is no obligation to report carrying $200 in a wallet.
- *Named parties to a specific note* are compellable to disclose that note's counterparty and surrounding context on warrant.
- *Pool-rate demurrage* makes long-term cash hoarding economically unviable, bounding the value of Notes-as-store-of-wealth for illicit actors.

The regime is strictly more privacy-preserving than the current surveilled-electronic-transfer regime (which exposes every edge to aggregators and KYC providers) and strictly more compliance-friendly than the current paper-cash regime (which exposes *no* issuance trail at all, regardless of who is investigating).

13.6 The CBDC Comparison

China's e-CNY, the European Central Bank's proposed digital euro, the Bank of Canada's CBDC research – all of these designs include either explicit or implicit surveillance capabilities. The central bank can see, freeze, expire, restrict, geofence, or retroactively examine any holder's balance.[2]

This is not an accident of the technology; it is a design goal. The pitch to governments is precisely that CBDCs give them visibility into civilian payment flows that paper cash denies them.

BUCK Notes is the opposite posture. Visibility is at the edges of a single instrument (the two named parties to a specific note), never across the system. The bearer-circulation phase of B1 notes is private in the way cash is private; the addressed A1/A2 phase is private between the two named parties the way a sealed letter is private. A jurisdiction adopting BUCK Notes gets better per-instrument compliance tools than paper cash provides today without enabling the population-scale surveillance a CBDC would.

14 Comparison With Prior Art

This section places BUCK Notes in the lineage of digital cash and privacy-preserving pool designs from 1982 to the present, and identifies what each predecessor lacked.

14.1 Chaum's E-Cash (1982-1998)

David Chaum's blind-signature scheme[4] introduced the core idea: an issuer (a bank) signs a token blinded by the holder so that the bank cannot link the issuance to the redemption. When the holder spends the token, the merchant verifies the signature without revealing who originally received it.

This is a beautiful design and operationally similar to BUCK Notes in spirit. Its failures were *institutional*, not cryptographic:

- Required a centralized issuer (the bank)
- Required online verification at spend time (no offline transferability)
- Two-sided market never bootstrapped (no merchants accepted, so no consumers held)
- DigiCash (the commercial implementation) went bankrupt in 1998

BUCK Notes has the same privacy properties without the centralized issuer (any identity-registered party can mint notes against their own on-chain BUCK). Spend-time verification is on chain rather than via the issuer (so the issuer cannot selectively block redemptions). The two-sided market problem is mitigated by BUCK's existing on-chain ecosystem – BUCK already exists as a usable currency, so adopting BUCK Notes is a UX upgrade rather than a greenfield bootstrapping.

14.2 Brands' Offline Divisible E-Cash (1993)

Stefan Brands' scheme[5] addressed the offline-spend problem: a divisible coin can be spent in pieces without contacting the issuer, but *double-spending mathematically reveals the spender's identity.*

This is the "deterrent rather than prevention" model. It works in theory but has two operational issues:

- The deterrent only deters if identity revelation has consequences (it relied on a legal/social enforcement layer external to the cryptography).
- The complexity of divisible-coin management was substantial and never produced a shipped product.

BUCK Notes handles double-spend by *prevention* (global nullifier set), not deterrence. No legal layer is required. Divisibility is achieved by issuing multiple notes at mint time rather than by a single divisible coin.

14.3 Casascius Coins (2011-2013)

Mike Caldwell's Casascius coins[3] were physical Bitcoin: a brass or silver coin with a holographic seal covering a private-key QR code. Snap the seal, scan the QR, the coin's Bitcoin balance was yours.

Casascius coins remain the closest existing analogue to paper BUCK Notes. What they lacked:

- Native integration with the underlying token (the Bitcoin protocol does not understand bearer-coin form; the coin is an aftermarket wrapper).
- Targeting (all coins were pure bearer; no way to issue a coin claimable only by a specific recipient).
- Demurrage (a Casascius coin holds its full Bitcoin balance forever, unrelated to any economic mechanism).
- Programmable predicates (only one mode: bearer).
- Issuer was a single individual, who eventually stopped issuing in 2013 due to FinCEN money-transmitter concerns – an issuance bottleneck that BUCK Notes distributes to all identity-registered participants.

BUCK Notes is, in essence, what Casascius would have been if Bitcoin had been designed with bearer-paper as a first-class instrument from the start, and if the underlying token had a demurrage and identity layer.

14.4 BIP-38 Paper Wallets

Encrypted Bitcoin private keys printed on paper, with a passphrase needed to spend. BIP-38 paper wallets are an aftermarket bearer instrument similar to Casascius but without the physical coin form factor.

Same limitations as Casascius (no native integration, no targeting, no demurrage, no predicates), plus the additional UX burden of remembering or transmitting the passphrase.

14.5 Zerocash / Zcash (2014-)

Ben-Sasson, Chiesa, Garman, Green, Miers, Tromer, and Virza's Zerocash[7] gave us the first scalable on-chain commitment-and-nullifier UTXO model with full zero-knowledge. Zcash deployed it in 2016.

Zcash provides on-chain shielded transactions but does *not* extend the model to physical bearer instruments, targeted cheques, or identity-linked issuance. All Zcash value lives on chain in shielded notes. No mechanism exists for printing a shielded Zcash note as a paper instrument that someone else could deposit later.

BUCK Notes uses Zerocash's commitment-and-nullifier primitive for its on-chain machinery and extends it with the Identity-binding layer and the physical-paper surface. The contribution is in the extension, not the primitive.

14.6 Aztec / Zk.Money

Aztec's predicate-encrypted notes[8] generalize Zerocash to private smart contract calls – a note can encode arbitrary state operated on by arbitrary predicates. This is the cryptographic ancestor of the `predicate` field in the BUCK Note commitment.

Aztec does not have physical bearer instruments either. Its predicates are designed for on-chain DeFi applications (private DEX trades, private lending), not for paper notes or identity-bound cheques. BUCK Notes applies the same predicate machinery to a different problem: specifying spend conditions for a private bearer instrument that is issued against an identity-registered reserve.

14.7 Tornado Cash and Privacy Pools

Tornado Cash[9] was the simplest possible commitment-and-nullifier mixer: fixed-denomination deposits in, anonymous withdrawals out. Sanctioned by OFAC in 2022 for money-laundering use.

Privacy Pools (Buterin, Schaefer, Tromer, et al., 2023)[10] extends Tornado with per-spend "association sets" – a depositor can prove their deposit is in a non-incriminating subset, allowing compliance without identification.

Both are mixer designs, not cash-instrument designs. Their relevance to BUCK Notes is the lineage: the Note Pool is a Tornado-style global anonymity set, but with predicates and Identity bound in. The selective-disclosure compliance story borrows from Privacy Pools' association-set ideas, but with issuer identity substituting for the depositor-declared non-incrimination proof.

14.8 Comparison Summary

System	Bearer paper	Targeting	Demurrage	Predicates	Native	Decentralized	Year
Chaum E-Cash	No	No	No	No	-	No	1982
Brands Offline	No	No	No	No	-	No	1993
BIP-38 Paper Wallet	Yes	No	No	No	No	Partial	2010
Casascius Coin	Yes	No	No	No	No	No	2011
Zerocash / Zcash	No	No	No	No	-	Yes	2014
Tornado Cash	No	No	No	No	-	Yes	2019
Aztec	No	No	No	Yes	-	Yes	2020
Privacy Pools	No	No	No	Limited	-	Yes	2023
CBDCs (e-CNY, etc.)	No	Yes	Sometimes	No	Yes	No	2020+
BUCK Notes	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	2026?

The "Native" column captures whether the bearer-paper instrument is a first-class designed primitive of the system (as opposed to an aftermarket wrapper or absent entirely). BUCK Notes is the first design that fills every column. This is not because the prior art was incompetent; it is because the four required pieces (commitment-pool primitive, identity layer, demurrage, jurisdictional reserve) finally converged in one system.

14.9 Comparison: Cash, CBDC, BUCK Notes

A more direct comparison aimed at a non-cryptographer reader: how does BUCK Notes stack up against physical cash and central bank digital currencies?

Property	Physical Cash	CBDC (typical)	BUCK Notes
PRIVACY			
Holder's identity hidden	Yes	No (issuer sees all)	Yes (during circulation)
Transaction graph hidden	Yes	No	Yes
Issuer's identity hidden	Yes (notes are fungible)	No	Only to bulk observers; visible bilaterally
FINALITY			
Settlement is instant	Yes	Yes	Yes (chain finality)
Reversible by issuer	No	Yes (CBDC freeze)	No
Reversible by holder	No	Yes (chargeback)	No
INFRASTRUCTURE			
Works without internet	Yes	No	Yes (transfer); needs net for deposit
Works without battery	Yes	No	Yes (paper); needs power for deposit
Works without bank account	Yes	No	Yes
Cross-border in your pocket	Yes (with declaration)	No (geofenced)	Yes
DURABILITY			
Survives infrastructure failure	Yes	No	Yes (paper); chain on recovery
Counterfeit-resistant	Partial (printed security)	Yes	Yes (cryptographic)
Recoverable if lost	No	Yes (issuer can re-issue)	No (but demurrage returns value to system)
ECONOMIC BEHAVIOR			
Lost notes extracted forever	Yes	No	No (returned via demurrage)
Inflation tracks issuer policy	Yes	Yes	Yes (tracks BuckK)
Holds value over decades	No (inflation)	No (inflation)	No (demurrage)
Holds value over years	Approximately	Approximately	Yes (with demurrage discount)
PROGRAMMABILITY			
Time-locked spends	No	Possible	Yes
Conditional payouts	No	Possible	Yes
Multi-signature spending	No	Possible	Yes
Issuer can restrict spending	No	Yes	No (issuer relinquishes control at mint)
COMPLIANCE			
Issuance is auditable	No	Yes	Yes
Circulation is auditable	No	Yes	No (private)
Deposit is auditable	Partial (CTR rules)	Yes	Yes
Selective disclosure available	No (no records)	No (full disclosure)	Yes

The dominant insight from this table: *BUCK Notes combines the privacy properties of physical cash with the compliance and durability properties of CBDC, while preserving infrastructure independence that CBDC abandons.*

The single property where physical cash is genuinely irreplaceable – "works without internet for deposits" – is moderated for BUCK Notes but not eliminated. Holders can transfer BUCK Notes hand-to-hand offline, indefinitely; they need internet only when they want to convert the bearer instrument to an on-chain balance. In a disaster scenario, BUCK Notes circulate as cash for the duration of the outage, then deposit normally when the chain becomes reachable again.

15 Required Identity Cryptography

BUCK Notes re-use the existing Identity machinery (Identity) without introducing new hardness assumptions. The whole construction is built from primitives already proved sound in the Proofs document. The only new moving parts are two standard SNARK-side primitives (Poseidon hash, Poseidon-based nullifier PRF) whose role is internal to the circuit.

15.1 Components Already Present

The following are unchanged from Identity v1 and are used directly:

1. **Pointcheval-Sanders issuer signature** σ on the Identity scalar m. Re-randomization continues to serve unlinkable credential generation (Identity Fountain). No change.
2. **ElGamal registered credential** $E = (R, C) = (rG, M + r \cdot pk)$ where $M = m \cdot G$. Continues to be the on-chain identity witness, re-randomized by the issuer at mint time to produce E_{note} bound to a specific recipient. No change.
3. **Chaum-Pedersen equality of discrete logs (Theorem 6).** Used verbatim by the A-spend predicate to prove that the depositor's *current* registered credential decrypts to the same `M` that E_{note} encrypts. This is the same proof the `approve` flow produces outside a SNARK; in Notes it is embedded in the spend circuit.
4. **Schnorr signature on** cm. Used in the mint circuit to bind A1 and B1 notes to their public issuer. Standard construction.
5. **Schnorr PoK of** sk **for registered** pk. Used by the B-spend predicate to bind the depositing account to a registered identity. Single-statement case of CP.
6. **IdentityRegistry** mapping between on-chain addresses and registered credentials (PS signatures, ElGamal ciphertexts, public-claim flags).

Notably absent: the stealth-address construction (Theorem 7) is *not* used by Notes. Recipient binding is handled directly by re-randomizing the recipient's registered ElGamal credential and verifying via CP-equality – the same mechanism Identity already uses to defeat the BUCK mutual-decryptability requirement.

15.2 New Primitive: The Note Commitment Hash

A SNARK-friendly commitment hash H_{cm} (Poseidon) is needed to compute the commitment inside the spend circuit. The hash is deterministic, collision-resistant over the note's field, and efficient in R1CS; Poseidon over the BN254 scalar field is the natural choice and is already used by the Aztec and Zcash Sapling circuits.

No novel cryptographic construction; only a circuit-level primitive that is standard in modern SNARK systems.

15.3 New Primitive: The Nullifier PRF

The nullifier PRF is Poseidon-based, keyed differently per spend shape:

- **A-spend:** key = sk_{dep}. Nullifier = $H_{\text{nf}}(\text{A} \| sk_{\text{dep}} \| \rho)$. By CP-equality, sk_{dep} is bound to M_{rec}; any credential attesting to the same `M` produces the same nullifier, so a recipient who re-issues their Identity with a fresh signing key still computes the same nullifier (key-loss recovery).
- **B-spend:** key = ρ itself; nullifier = $H_{\text{nf}}(\text{B} \| \rho)$, publicly computable by anyone who knows ρ. This is the first-come-first-served property of bearer B1 notes.

A domain-separator byte (A vs. B) in the PRF input keeps the nullifier namespaces disjoint across flavors.

15.4 Re-Encryption at Mint (A1 and A2)

The issuer looks up the recipient's registered credential $E_r = (R_r, C_r)$ in IdentityRegistry, picks fresh $r' \leftarrow \mathbb{Z}_q^*$, and computes

$$E_{\text{note}} \;=\; (R_r + r'G,\; C_r + r' \cdot pk_{\text{rec}}) \;=\; ((r_r + r')G,\; M_{\text{rec}} + (r_r + r')\, pk_{\text{rec}}).$$

This is a single additional ElGamal re-randomization under the recipient's existing pk_{rec}. The issuer does *not* learn any additional secret; they compute E_{note} from public data (E_r, pk_{rec}) and their own chosen r'. Single curve multiplications on BN254 G_1. No pairing, no new hardness assumption.

For A1, the issuer additionally signs cm under their own sk_{issuer} and embeds $(m_{\text{issuer}}, \sigma)$ in the payload. For A2, no such signature; the issuer's `M` is instead carried as an encryption for the recipient to satisfy the mutual-decryptability invariant.

15.5 CP-Equality at Spend (A-Spend)

The depositor produces a CP-equality proof (Theorem 6) showing that E_{note} and the depositor's *currently registered* credential $E_{\text{dep}}^{(\text{reg})}$ decrypt to the same `M`. The extractable witness is sk_{dep}; because sk_{dep} is bound (by registration) to `M`, the proof establishes the depositor owns the registered credential encrypting the recipient's `M`.

This is identical in structure to the CP proof Alice produces during `approve`, lifted into the SNARK circuit rather than verified on-chain as a Fiat-Shamir NIZK. No new assumption.

15.6 Schnorr PoK at Spend (B-Spend)

A B-spender produces a Schnorr PoK of sk_{dep} for their registered pk_{dep}, Fiat-Shamir bound to the spend transaction. This anchors the deposit in a registered Identity (so KYC/FINTRAC obligations attach) while leaving the deposit open to any registered holder of ρ. Standard Schnorr; no novelty.

15.7 Summary: New Cryptographic Components

Component	Status	Assumption
Poseidon commitment hash	New circuit primitive (standard in SNARKs)	ROM-like
Poseidon nullifier PRF	New circuit primitive	ROM-like
PS signature (issuer)	Re-used from Identity	q-SDH
ElGamal credential + re-encryption	Re-used from Identity	DLog
Chaum-Pedersen equality (Theorem 6)	Re-used (lifted into SNARK) from Identity	DLog + ROM
Schnorr signature + PoK of sk	Re-used from Identity	DLog + ROM
SNARK proof system (Groth16/PLONK)	Standard on BN254	system-dep

Only the Poseidon hash family is new, and it is already standard in the SNARK ecosystem. No novel hardness assumption is introduced. Theorem 7 (stealth-address) remains available in the Proofs document as an independent primitive but is not invoked by the Notes construction.

16 The SNARK Circuits

Because the flavor taxonomy collapses to two spend-circuit shapes, the Notes system requires exactly three circuits: one *mint* and two *spend* (A-spend for addressed A1/A2 deposits, B-spend for bearer B1 deposits). A single universal spend circuit dispatched on a public `flavor` input is a valid deployment variant; the trade-off is a larger verifier vs. fewer distinct trusted-setup ceremonies.

16.1 Mint Circuit

Public inputs. `totalFace` (the total face value $\sum v_i$ of the notes being minted), $\{cm_1, \ldots, cm_N\}$ (commitments to be appended), and `issuerRegistryEntry` (a Merkle root or index into IdentityRegistry attesting to the issuer's registered credential).

Witnesses. For each note i: $(\text{flavor}_i, v_i, \rho_i, \text{id-payload}_i, \text{predicate}_i)$, plus the issuer's private material required by the flavor.

Constraints.

1. $\sum v_i =$ `totalFace` (amount conservation).
2. Each $cm_i = H_{\text{cm}}(\text{flavor}_i, v_i, \rho_i, \text{id-payload}_i, \text{predicate}_i)$ (opening binding).
3. For each **A2** note (private issuer, addressed): the issuer re-randomizes the recipient's registry credential $E_r = (R_r, C_r)$ with fresh r'_i: $\text{id-payload}_i = E_{\text{note},i} = (R_r + r'_i G,\ C_r + r'_i\, pk_{\text{rec},i})$. The issuer's registered ElGamal decryption of its own credential is bound into the commitment via M_{issuer} (not revealed), to keep the standard BUCK mutual-decryptability payload $E_{\text{issuer-for-rec}}$ available at deposit time.
4. For each **A1** note (public issuer, addressed): same as A2 plus the payload contains plaintext m_{issuer} and a Schnorr signature σ_i on cm_i under $pk_{\text{issuer}} = m_{\text{issuer}} G$.
5. For each **B1** note (public issuer, bearer): the payload is $(m_{\text{issuer}}, \sigma_i)$ with σ_i a valid Schnorr signature on cm_i under pk_{issuer}. No recipient binding.

Contract side. The contract verifies the proof, pulls `totalFace` BUCK from the caller (ordinary ERC-20 transfer; the caller pays outgoing-transfer demurrage), appends each cm_i to `noteTree`, and increments `noteFaceSum` by `totalFace`. No demurrage arithmetic lives in the SNARK.

16.2 A-Spend Circuit (Addressed: A1 / A2)

This circuit handles both addressed flavors. A1 and A2 differ only in whether the commitment payload carries $(m_{\text{issuer}}, \sigma)$ alongside E_{note}; the recipient-binding mechanism – CP-equality against the depositor's *current* registered credential for M_{rec} – is identical.

Public inputs. `noteRoot`, `nullifier`, `faceValue`, `depositorRegistryEntry`.

Witnesses. $(\text{flavor}, v, \rho, \text{id-payload}, \text{path})$, sk_{dep}, the Merkle path to the depositor's current registry credential $E^{(\text{reg})}_{\text{dep}} = (R^{\text{reg}}, C^{\text{reg}})$.

Constraints.

1. Merkle inclusion of cm under `path` in `noteTree`.
2. `nullifier` $= H_{\text{nf}}(\text{A} \| sk_{\text{dep}} \| \rho)$.

3. The payload contains $E_{\text{note}} = (R_n, C_n)$; the depositor's registered credential $E_{\text{dep}}^{(\text{reg})}$ is present in IdentityRegistry; and CP-equality holds: $C_n - sk_{\text{dep}}\, R_n = C^{\text{reg}} - sk_{\text{dep}}\, R^{\text{reg}}$ (Theorem 6). This proves the depositor's Identity `M` equals the recipient's `M` bound at mint, without revealing either.

4. If the flavor is A1: the payload also contains $(m_{\text{issuer}}, \sigma)$; verify σ under $pk_{\text{issuer}} = m_{\text{issuer}}G$ on cm. (A2 omits this constraint; no public issuer to verify.)

5. `faceValue` $= v$.

The contract checks the nullifier is unused, reads its own (B, A), performs the age-preserving transfer of v balance and $v \cdot A/B$ BUCK-age from pool to depositor, and decrements `noteFaceSum` by v.

Key-loss recovery. The binding is to `M`, not to the specific pk_{rec} used at mint. A recipient who rotates their signing key via IdentityRegistry re-issuance still satisfies the CP-equality check with their new credential.

16.3 B-Spend Circuit (Bearer: B1)

Public inputs. `noteRoot`, `nullifier`, `faceValue`, `issuerPkClaim` (plaintext pk_{issuer} for wallet display, redundantly verified by the circuit), `depositorRegistryEntry`.

Witnesses. $(v, \rho, \text{id-payload}, \text{path})$, sk_{dep}, Schnorr nonces.

Constraints.

1. Merkle inclusion of cm under `path`.

2. `nullifier` $= H_{\text{nf}}(\text{B} \| \rho)$. Derivable from ρ alone – this is the first-come-first-served property of B1.

3. The payload contains $(m_{\text{issuer}}, \sigma)$; verify σ on cm under $pk_{\text{issuer}} = m_{\text{issuer}}G$ in IdentityRegistry.

4. Schnorr PoK of sk_{dep} for pk_{dep} in IdentityRegistry, Fiat-Shamir bound to `msg.sender`, `chainid`, `nullifier`.

5. `faceValue` $= v$.

Contract delivery is identical to A-spend: age-preserving transfer of v balance and $v \cdot A/B$ BUCK-age from pool to depositor, computed outside the SNARK.

16.4 Circuit Engineering Budget

Ballpark constraint counts (orders of magnitude, BN254 Groth16):

Circuit element	Approx. constraints
Poseidon commitment hash (1 call)	~250
Poseidon nullifier PRF (1 call)	~250
Merkle path verification (20 levels)	~5000
ElGamal / Chaum-Pedersen (A-spend)	~2000
Schnorr signature verify (A1, B1)	~500
Schnorr PoK (B-spend depositor)	~500
Range check on v	~100

A-spend totals ~8,000 constraints (includes the A1 signature verify); B-spend totals ~6,000. Comparable to Zcash Sapling's Spend circuit (~25,000, with randomness features we don't need) and Aztec's basic note-spend circuit. On-chip proving time on a modern smartphone: a few seconds.

16.5 Common Circuit Gadgets

Shared library across all three circuits:

- Merkle inclusion (Poseidon path)
- Poseidon hash wrapper
- ElGamal re-randomization and CP-equality
- Schnorr signature verify + Schnorr PoK
- Range proofs on field elements (for v)

Flavor-specific wiring layers on top. Standard SNARK library architecture (Aztec Noir, Circom, Halo2 all support this layering).

17 What This Architecture Rules Out: Pure-Bearer Cryptographic Notes

The design intentionally does not provide a pure-bearer cryptographic note – one whose spend predicate is "anyone who knows the opening can deposit, with no Identity hook at all." This is the closest analogue to a $20 bill (possession is the entirety of ownership, no registration involved), and it is attractive for the same reasons physical cash is attractive. Three properties make it a poor fit for the cryptographic surface:

1. *Issuer-race is unbounded.* A pure-bearer predicate knowρ has no cryptographic mechanism preventing a malicious issuer from retaining ρ and racing the holder to deposit. The physical seal on paper makes the *observation* of ρ detectable to the holder, but does not constrain the *issuer* at mint time.

2. *Deposit has no Identity binding.* BUCK's mutual-decryptability invariant and the regulatory audit path both rely on the depositor being a registered Identity. A pure-bearer deposit with no Identity hook breaks this property and forces the system into one of two uncomfortable corners: accept unregistered deposits (losing all compliance), or reintroduce Identity at deposit time (at which point it is just a B1 note).

3. *Three alternative routes all collapse into existing flavors. Blind minting* (the issuer does not know ρ at mint time) requires the holder to participate in minting with a blinded commitment – incompatible with "print a stack of notes and hand them out." *Trusted printer* (HSM-attested erasure of ρ) works in principle but reintroduces a single-point-of-trust. *Short expiry* limits the issuer-race window – equivalent to B1 with an additional expiry predicate.

Rather than contort the cryptographic layer to approximate pure bearer, BUCK Notes keeps the physical-world primitive where it belongs: the *tamper-evident seal over a B1 note* is a physical-exclusivity mechanism, and that is an inherently physical property. Cryptography makes the *opening* hard to forge; a hologram makes the *observation* detectable. The two surfaces complement without pretending to be the same primitive.

For use cases that truly require "no identity link at all":

- *Short-lived hand-to-hand circulation* is served by B1 paper notes with tamper-evident seals, printed by a trusted issuer who publicly warrants that they erased ρ after printing. This warranty is an off-chain social/legal claim identical in kind to the trust extended to the Bank of Canada when it issues physical polymer banknotes. A credit union or a commercial issuer fills the role of the central bank for this purpose.

- *Long-term holding* is served by addressed flavors (A1, A2), where CP-equality binds the note to the recipient's `M` and no hologram seal is needed.

The canonical "walking-around money" digital flow is handled by A2 notes: Alice targets the payee's registered `M` at the moment of payment, hands over the opening via message or QR, and the recipient deposits into their own account. This changes the UX relative to paper cash – payment now involves a brief digital handshake – but gains a property physical cash lacks: a dropped or intercepted A2 note is useless to anyone but the intended recipient.

18 Security Assumptions and Theorems Used

The unified Notes design rests on the same cryptographic foundations as the existing Identity system, plus one already-proved additional theorem:

Primitive / Theorem	Usage in Notes	Assumption
PS signature (issuer credential)	Mint-circuit issuer-binding (all)	q-SDH
ElGamal on G_1	Recipient payload (A1, A2)	DLog on G_1
Theorem 6 (CP re-encryption)	A-spend predicate	DLog + ROM
Schnorr signature verify	Public-issuer binding (A1, B1)	DLog + ROM
Schnorr PoK (sigma protocols)	B-spend depositor binding	DLog + ROM
Poseidon hash/PRF (SNARK-side)	Commitments, nullifiers	ROM-like (circuit)
SNARK proof system (e.g., PLONK)	Wrapping all of the above	system-specific
Merkle tree (noteTree)	Commitment anchor	collision-resistance

No *novel* cryptographic assumption is introduced. The only new primitive not already in BUCK is the Poseidon hash family, which is standard in SNARK deployments (Aztec, Zcash Sapling/Orchard, Starknet, Mina) and well-studied.

18.1 Demurrage Conservation as a Contract Invariant

Because the pool is a regular BUCK account and redemption uses a two-sided-balanced age-preserving transfer, no Notes-specific demurrage invariant is needed. The system-wide BUCK invariants

$$\sum_{\text{all accounts}} B_X \;+\; B_{\text{Jubilee}} \;=\; \text{const}, \qquad \frac{d}{dt}\sum_{\text{all accounts}} A_X \;=\; \sum_{\text{all accounts}} B_X$$

hold trivially through redemption: the v balance and $v \cdot A/B$ age moved from pool to recipient appear on both sides of each sum. The Notes contract only needs to maintain two auxiliary local invariants:

1. $F \geq 0$ (no negative note face sum).
2. Every v appearing in a deposit equals the v that appeared in the corresponding mint (enforced by the SNARK via the Merkle inclusion of cm binding v).

Both are trivial. The only new mechanism is the age-preserving transfer primitive, itself a simple state-machine operation on two accounts' (B, A) pairs.

19 Implementation Scope

Rough engineering estimate, assuming the existing Alberta Buck contract skeleton and an existing SNARK toolchain (Aztec Noir or Circom/snarkJS):

19.1 Smart Contract Additions

- `noteTree` (Merkle tree with Poseidon nodes): ~1 week, bulk of code is existing Aztec/Zcash sapling implementations.
- `nullifiers` mapping: trivial.
- `noteFaceSum` scalar + mint/deposit updates: trivial (audit only).
- Age-preserving transfer primitive (system-level hook writing to two accounts' (B, A) atomically): ~1 week.
- A-spend and B-spend verifiers (Groth16-style on-chain): ~1 week integration per circuit.
- Mint-circuit verifier: ~1 week.

Total: ~6 weeks for an experienced Solidity engineer.

19.2 SNARK Circuits

- Mint circuit: ~2-3 weeks (Poseidon commitments, ElGamal re-randomization, Schnorr signature verification inside the circuit).
- A-spend and B-spend circuits (with shared gadgets): ~3-5 weeks.
- Common gadgets library (Merkle, Poseidon wrappers, ElGamal/CP, Schnorr, range checks): ~2 weeks.

Total: ~8 weeks for an experienced SNARK engineer.

19.3 Wallet & UX

- Recipient scan loop: trial-decrypt A1/A2 payloads against registry credential; detect B1 broadcasts: ~1 week.
- A2 self-wormhole mint flow (issuer = recipient): ~1 week.
- A1 identified-cheque creation and deposit UX: ~2 weeks.
- A2 addressed-cheque creation and deposit UX: ~1 week (reuses A1 code paths).
- B1 bearer-cheque creation, sealed-QR print flow, redemption UX: ~2 weeks.
- QR-print templates, hologram-seal vendor integration: ~1 week.
- Prover service integration for thin clients: ~2-3 weeks.

Total: ~9 weeks for a wallet team.

19.4 Cross-Checks and Audit

- Cryptographic review of the spend circuits: 4 weeks, external auditor with SNARK-circuit expertise.
- Economic model validation (pool-rate demurrage in stress scenarios: low turnover, high turnover, mint/deposit mismatch, large accumulated lost-note face sum): 2 weeks.
- End-to-end fuzz and property-based testing: 2 weeks.

Total: ~8 weeks of external + internal audit.

19.5 Phased Rollout

The engineering work is naturally phaseable:

Phase	Scope	Dependencies
1	On-chain note pool: `noteTree`, nullifiers, mint-verifier	Existing BUCK ERC-20 deployment
2	A-spend circuit (A2 self-wormhole + addressed cheque UX)	Phase 1, Identity CP gadget
3	A1 public-issuer addressed cheque (signature-in-payload)	Phase 2
4	B-spend circuit + B1 bearer cheque + sealed-QR print flow	Phase 1, hologram vendor
5	Programmable `predicate` field (bounded predicate DSL)	Phases 1-4, Aztec Noir toolchain
6	Optional: off-chain re-targeting via proxy re-encryption	Phases 1-4, Identity Guardians

Phase 1-2 establishes the minimum viable Notes system (private addressed cheques + self-wormhole). Phase 3 adds auditable issuance (payroll, B2C). Phase 4 adds bearer instruments. Phases 5-6 are optional extensions deferred until the core system is production-proven.

Overall program estimate: 5-7 months of focused work to ship Phases 1-4 as a production-quality BUCK Notes system. Phases 5-6 add 4-8 months of further engineering if demand materialises.

19.6 Regulatory and Legal Engagement

In parallel with engineering, the legal and regulatory work:

- Confirm that BUCK Notes falls within Alberta's provincial jurisdiction over property and civil rights (s 92(13)) – the analysis in the Legal Foundation document covers the underlying primitive; bearer notes are an extension that may warrant additional treatment.
- Engage FINTRAC on the issuance-side reporting model.
- Engage the Office of the Superintendent of Financial Institutions (OSFI) on reserve treatment.
- Coordinate with Alberta Treasury Branches (ATB) as a candidate first issuer of paper notes under a public-warranty of ρ-erasure for the B1 walking-around-cash surface.

20 Open Questions

The architecture above leaves a handful of economic and cryptographic questions worth sharpening before commitment:

20.1 Donation Incidence

Donations to the contract lower A/B and therefore lower the age-share every subsequent redeemer absorbs. The benefit spreads across all future redemptions rather than accruing to a single lucky first-redeemer. Because the age-share calculation uses the *current* A/B at each redemption, a donation immediately before a large redemption benefits that redemption more than a small one – but every subsequent redemption benefits in proportion. This is reasonable behaviour without any special smoothing logic.

Residual donation principal (after all notes drain) is not lost: it remains in B with $A = 0$, available to subsidise the next generation of mints (whose recipients will absorb an artificially low A/B at their redemption).

20.2 Stranded Value on Lost Notes

A lost note contributes to F indefinitely. The BUCK backing that note sits in the pool, accumulating BUCK-age. Since the age-preserving transfer uses A/B (not A/F), the lost face value weighs on A through continued age accumulation, and B remains unclaimed. Redemptions of live notes absorb an ever-larger A/B as the lost share's age dominates. At 1%/yr loss and 2%/yr demurrage, the long-term tendency is that the lost note's implicit demurrage is paid instalment-style by subsequent live noteholders.

A reclaim path is possible: after a very long period (say 99 years), the contract could allow the issuer to submit a zero-knowledge proof that a specific note commitment remains unspent, and reclaim the residual principal less accrued demurrage. Requires issuer-side recordkeeping; deferred.

20.3 Integer-Arithmetic Rounding

Redemption involves two integer divisions (computing $v \cdot A/B$ for the age-share). Any residual from truncated division remains in the pool's (B, A) state, raising A/B slightly for the next redeemer – the rounding is self-healing rather than drift-accumulating. Choice of rounding direction (floor vs. banker's) is a minor contract-level decision with no cryptographic impact.

20.4 Predicate Programmability vs. Circuit Simplicity

Keeping three fixed flavors simplifies the circuit count. Adding programmable predicates (time locks, oracle conditions, multi-sigs) re-introduces most of the complexity that the fixed flavor scheme was chosen to avoid. A compromise: support a *bounded* predicate language in the circuit (time-lock + oracle-condition + up-to-3-sig) rather than a fully general one. The bounded language covers 90%+ of the programmable-predicate use cases while keeping the verifier compact.

20.5 Universal Spend Circuit vs. Two

Deploying two distinct spend verifiers (A-spend and B-spend) costs gas per transaction (one verifier per call). A universal verifier that takes `flavor` as a public input and branches internally costs a larger but single verifier. Gas trade-off depends on production transaction mix; if A-flavor

dominates, a dedicated A-spend verifier is cheaper. Empirical question, resolvable at deployment time.

20.6 Regulatory Interaction with the Age-Preserving Transfer

Jurisdictional regulators (FINTRAC, OSFI) have rules around bearer instruments and tax treatment of negative-interest accounts. BUCK's on-chain demurrage was argued through separately, and the Notes contract inherits that treatment with no pool-specific novelty – the contract is just another BUCK account.

The one new wrinkle is the *age-preserving transfer* itself: ordinary BUCK transfers carry only balance (recipient gets "fresh" BUCK), whereas Note redemptions carry balance plus a proportional BUCK-age. This means a Note recipient's post-redemption spendable value is less than their nominal balance – the difference being the demurrage they will pay on their next outgoing transfer. For tax and reporting purposes, whether the "face" or the "spendable" number should be used as the income figure is a question for tax counsel. The on-chain state is fully transparent either way.

21 Summary

BUCK Notes is the Alberta Buck's native private-bearer primitive. A single Merkle tree of commitments, a single nullifier set, and three realisable flavors – *A2* (addressed, private issuer; also self-wormhole), *A1* (addressed, public issuer; payroll and audited cheques), and *B1* (bearer, public issuer; cashier's cheques) – span every use case the Alberta Buck series needs a bearer instrument to cover, from cold-storage cryptographic self-wormholes to walking-around paper notes to cross-border targeted remittances. The fourth cell, private-bearer, is ruled out by BUCK's mutual-decryptability invariant.

Demurrage needs no note-side machinery: the Note Pool is literally the BUCK ERC-20 contract's own balance – a regular BUCK account with its own B and A in the ERC-20 ledger. The only Note-side state beyond the Merkle tree and nullifier set is one scalar F, the total face value of live notes, kept as an audit quantity. Redemption delivers the *full face value* v to the recipient, together with the pool's proportional share of BUCK-age $v \cdot A/B$; the recipient absorbs this age onto their own account and bears the demurrage cost on their next outgoing transfer through the ordinary BUCK accounting path. This is exactly the demurrage they would have paid had they held v BUCK personally through the pool's residence period. The entire mechanism requires a single new primitive, an age-preserving transfer from pool to recipient, localized to the Notes contract's one egress function. Donations to the pool dilute A/B and cheaply subsidise all subsequent redeemers. Conservation of balance, age, and Jubilee growth all fall out exactly.

The addressed-versus-bearer distinction produces a sharp partition of which flavors are safe for long-term holding: *A1* and *A2* are cryptographically safe (CP-equality binds the note to the recipient's `M`; Identity re-issuance recovers from key loss). *B1* is a civil-trust instrument: the issuer necessarily knows ρ, so B1 cheques are meant to be cashed promptly and backed by the issuer's legal standing rather than by cryptography alone. This is an intrinsic property of the bearer predicate, not a policy choice.

The cryptographic architecture does not attempt to provide a pure-bearer note with no Identity hook. That use case remains the domain of physical trust models: tamper-evident seals over B1 paper notes, issued by a trusted printer with a public warranty that ρ was erased after printing. This keeps the cryptographic/physical boundary explicit rather than contorting the cryptographic layer to approximate a property that inherently requires physical primitives.

A BUCK Note can live on chain as a commitment-opening pair, travel between wallets as a digital message, or be printed on paper as a QR-encoded slip with a hologram seal (for B1) or without one (for A1, A2). All three surfaces share the same cryptography and the same on-chain footprint. Counterfeiting is cryptographically impossible: a "fake" note's commitment is simply absent from the on-chain tree, the Merkle proof fails, and the wallet displays `FAKE`. Lost notes are not extracted as seigniorage: the pool-age mechanism returns their value to the commons via demurrage pressure on subsequent live-note redemptions.

Issuance is on chain and identity-bound. Circulation breaks on-chain bulk traceability (no observable edge between payer and recipient) while preserving bilateral disclosure (both named parties always learn each other's `M`; either is compellable on warrant). Deposit is identity-bound and auditable. This is stronger compliance than physical cash provides today, and stronger privacy than any CBDC has ever proposed.

No novel cryptographic assumption is introduced. Theorem 6 of the Proofs document (Proofs) applies verbatim to the A-spend predicate (CP-equality on the recipient's re-randomized credential); B-spend reduces to Schnorr signature verification plus a Schnorr PoK of the depositor's registered secret. The only new circuit-level primitive is Poseidon, which is already standard across the SNARK ecosystem. Implementation scope for Phases 1-4 (the full A1/A2/B1 production surface)

is 5-7 months for a focused team.

BUCK Notes answers the question Chaum was asking in 1982: what does digital cash look like when you take "cash" seriously? The answer, forty years later: it looks like a commitment in an on-chain pool, openable as a message, a QR, or a printed slip, with a value that decays at the same rate your bank balance does, integrated with a complete monetary system that does not require a central bank to function. Alberta has, uniquely in the world, all four pieces required to build this. Whether to assemble them is, as ever, the only question that matters.