Accountable Anonymity for Public Money (DRAFT v0.1)

Identity-Bound Bearer Payments with No Trusted Opener

Perry Kundert

2026-06-11

Public money is going digital, and every design on the table pays for the convenience with a ledger its issuer can read, freeze, and mine. Cash resists surveillance, but only by keeping no evidence at all: no proof of payment, no recourse, no trail. This paper presents BUCK Notes, a bearer-payment layer for an identity-aware stablecoin that refuses both defaults. No one can trace payments in bulk – not an analytics firm, not an omnibus subpoena, not even the system's own KYC issuer. Yet the two parties to any payment can always prove exactly who the other was, with evidence sound enough to convict on. The guarantee is the **non-deniable receipt**: even a *colluding* payer and payee cannot complete a payment without each holding evidence that names the other.

The consequences read like due process implemented in mathematics (the **compulsion calculus**): evidence of every payment exists; only the participants hold it; each may surrender, refuse, or destroy their copy; and no institution can be compelled to produce what it provably never had. One design choice makes this work: payments bind KYC-certified *identities*, not just account keys (**Identity-M binding**). A single re-encryption gadget, instantiated three ways, yields every realisable note flavour (addressed or bearer, public or private issuer), all shipped and measured on Ethereum. We prove the fourth flavour impossible, the obvious cheap binding vacuous, and zero-knowledge registry membership necessary for the doubly-anonymous flavour under a stated restriction; exactly the branch the shipped system takes. Every theorem carries an executable witness. (PDF, Text)

Contents

1 Introduction

The choice a society makes about the privacy of its money is a choice about the balance of power between the individual and the institutions that watch over them. This paper is about a way to make that choice deliberately: payments no institution can read in bulk, where the two parties (and only they) always hold court-grade proof of who dealt with whom.

That choice is currently being made by default. Every central-bank digital currency in design or deployment couples digital convenience to a ledger some institution reads in bulk; the visibility is not an engineering accident; it is the pitch to the governments funding it[1]. Cash is the only widely used instrument that resists bulk observation, and it resists by carrying no information at all: no issuance trail, no proof a court can rely on, no recourse when stolen. The public debate treats these as the only two options: total legibility or total opacity. Forty years of cryptography has not yet dislodged that false dichotomy from policy.

Chaum asked the right question in 1982[2]; his answer stalled on institutional grounds, not cryptographic ones. The anonymity line that followed (Zerocash and its descendants, the mixers, the privacy pools[3, 4, 5]) made payments untraceable and unaccountable in the same stroke. The auditable-CBDC line that answered it (PRCash, UTT, PEReDi, Platypus[6, 7, 8, 9]) restored accountability by handing a designated authority the power to open transactions: an auditor, a committee, the central bank itself. No prior system provides both untraceable payments and mandatory accountability with *no opener at all*.

This paper presents a system that does, and locates precisely what it costs. The design goal is civic before it is cryptographic:

- **Justice must work.** When a payment is implicated in a crime, the legal system must be able to obtain cryptographically sound evidence of exactly who paid whom (strong enough to convict on, impossible to forge against the innocent). And the evidence must be guaranteed to *exist*: two colluding criminals must not be able to arrange an untraceable payment.

- **Compulsion must be targeted.** That evidence is held by (and can be demanded from) only the people party to the payment. There is no registry, exchange, issuer, or validator that a warrant, a breach, or a purchase can turn into bulk surveillance, because no such party holds the information in any usable form.

- **Refusal must be possible.** If the legal order itself turns unjust (an authoritarian government criminalising beliefs, associations, donations), a citizen retains the physical ability to refuse: to withhold, or irreversibly destroy, their own keys. The data then stays private against any coalition of compelled institutions and archived chain history. Surveillance of the unwilling is

[1] Bank for International Settlements. "Rise of the Central Bank Digital Currencies." [TODO]

[2] Chaum, D. "Blind Signatures for Untraceable Payments." CRYPTO 1982. [TODO exact cite]

[3] Ben-Sasson, E., et al. "Zerocash: Decentralized Anonymous Payments from Bitcoin." IEEE S&P 2014. [TODO]

[4] Pertsev, A., Semenov, R., Storm, R. "Tornado Cash." 2019. [TODO]

[5] Buterin, V., Schaefer, J., Tromer, E., et al. "Blockchain Privacy and Regulatory Compliance: Towards a Practical Equilibrium (Privacy Pools)." 2023. [TODO]

[6] Wuest, K., Kostiainen, K., Capkun, V., Capkun, S. "PRCash: Fast, Private and Regulated Transactions for Digital Currencies." FC 2019. [TODO verify authors/title]

[7] Tomescu, A., et al. "UTT: Decentralized Ecash with Accountable Privacy." IEEE S&P 2022. [TODO verify]

[8] Kiayias, A., Kohlweiss, M., Sarencheh, A. "PEReDi: Privacy-Enhanced, Regulated and Distributed Central Bank Digital Currencies." ACM CCS 2022. [TODO verify]

[9] Wuest, K., Kostiainen, K., Delius, N., Capkun, S. "Platypus: A Central Bank Digital Currency with Unlinkable Transactions and Privacy-Preserving Regulation." ACM CCS 2022. [TODO verify]

not available at any institutional price; it requires coercing human beings, one at a time, in the open.

These three goals are usually presumed to be in tension. The central claim of this paper is that they are simultaneously satisfiable, practically, on a public blockchain today.

What makes all three possible is one *counter-intuitive inversion*: payments must bind *identities* – certified once, at KYC, and held by persons – not just the *account keys* that hold funds. We call this **Identity-M binding**. Everything else follows from it: the receipt that always exists, key recovery without any custodian, the impossibility of one whole quadrant of the design space, and the exact place where accountability stops being cheap.

The unexpected result of adding *more* identity binding to the blockchain is that *less* of our identity leaks, and *more* of our rights and privacies are protected.

1.1 Contributions

Seven contributions, in the order the paper builds them:

- **The non-deniable-receipt invariant** (Def. 2): a security definition for payment systems whose new clause is collusion resistance: even a *willing* payer and payee cannot construct an accepted payment with no receipt. Accountability is **bilateral**, held by the counterparties; there is no system-wide opener.

- **The compulsion calculus** (Sec. 4.4): the invariant, composed with issuer-blindness (the KYC issuer cannot trace even its own certified clients; Thm. 3), yields three properties we believe are jointly new for a payment system: evidence *exists* for every payment; *only* the direct participants hold it; and it is *refusable*: without a participant's keys, surrendered or compelled, the ledger is computationally silent forever. We argue this is the right cryptographic substrate for due process, and the right failure mode under unjust process.

- **Identity-M binding** (Sec. 5): the inversion that makes the rest practical. One gadget (prove a re-encryption of a registered Identity under a target identity point) instantiated three ways covers the whole realisable design space. Note spend authorization keys on identity, so a lost account key is recoverable with no custodian, and the binding survives the time gap between an anonymous payer and an anonymous payee.

- **A complete, shipped construction** of all three realisable flavours – addressed/public (A1), addressed/private (A2), bearer/public (B1) – on Ethereum: Groth16 batch mint, flavour-agnostic spend, and spend-time composition of a deposit sigma, a bound set-membership proof, and a note-to-ciphertext tie, every verifier real and measured (Sec. 10).

- **The forced taxonomy** (Thm. 8): the fourth flavour – a bearer note from a hidden issuer – is impossible, by an information-theoretic argument.

- **The cost of anonymity, located** (Sec. 9): the receipt is nearly free when one party is named at the binding moment. When both are anonymous, the obvious cheap coupling is *vacuous* (Lem. 15), and a collusion-resistant receipt requires zero-knowledge registry set-membership under a single-binding-step restriction (Prop. 16). The shipped system takes exactly the branch the proposition permits, and we measure it.

- **A reproducible artifact**: contracts, circuits, and verifiers, open and test-covered – over 400 Foundry tests, 330 wallet tests, an end-to-end suite where every verifier is the real Groth16 artifact – and the paper's own claims carry executable witnesses (Appendix 12) that re-run on each build.

2 The Shape of the System, in Plain Language

Before the formalism, the experience.

Alice owes Bob one hundred BUCKs. She opens her wallet and mints a *note*: her wallet burns one hundred of her BUCKs into a shared pool and publishes a single opaque number (a cryptographic commitment) onto the public chain. The commitment reveals nothing: not the amount, not Bob, not (for the private flavour) even that it was Alice. Alice then hands Bob the note's *opening* (a few hundred bytes) by any channel she likes: a message, a QR code on her phone's screen, a slip of paper. Days or months later, from any account he controls, Bob *deposits* the opening: his wallet proves (in *zero knowledge*, revealing nothing about which note) that some unspent note in the pool is his to claim, and the pool pays him one hundred BUCKs. An observer watching the chain sees a commitment appear at one moment and an unrelated-looking payout occur at another, with nothing connecting them – not the amounts, not the parties, not the timing.

Three further facts make this different from a mixer (a pool that breaks payment trails and does nothing else), and they are the subject of this paper.

First, the payment writes its own sealed receipt. Built into every note (enforced by the chain's verifiers, impossible to omit) is the cryptographic material from which each party can produce a proof naming the *other party's certified legal identity*. Bob can prove "this payment came from Alice"; Alice can prove "Bob is the one who deposited it." The proof is sound: nobody can manufacture a receipt naming someone who was not actually party to the payment. And it is *unavoidable*: Theorem 12 shows that even if Alice and Bob conspire – both willing, both technically sophisticated – they cannot construct an accepted payment whose receipt material is missing or unusable. The worst a conspiracy can do is render a note addressed to nobody, and such a note is *unspendable*: the pool will not pay out on it.

Second, only Alice and Bob hold the seal. The receipt is not deposited with any registrar. It does not exist in any database. The KYC office that once certified Alice's identity cannot reconstruct her payments; we prove (Theorem 3) that even a *fully breached* identity office, holding every certificate it ever issued plus the entire blockchain, learns nothing about who transacted with whom. A court that needs the receipt has exactly one place to get it: from Alice, or from Bob, by lawful order directed at that person.

Third, the seal answers to its holders – and only its holders. Served with a lawful order, Bob can comply: he surrenders his receipt, and the court obtains evidence strong enough to convict on, evidence that mathematics – not testimony – ties to Alice. This is how the system serves justice: the victim of a fraud, the honest counterparty of a criminal, can always prove exactly whom they dealt with. But if the order is the instrument of an unjust regime – if the "crime" is a donation to a dissident, membership in a church, a purchase the state has decided to punish retroactively – Bob can refuse. He can decline to produce his key, accepting the personal legal consequence of that refusal; or he can have destroyed it long before, after verifying the payment, as a standing practice. Then *no one on earth* can open that payment: not the identity office, not the chain's operators, not a seizure of every server the system runs on. The mathematics does not know who is in power.

This is, we will argue, the right shape for the money of a free society: per-payment accountability *stronger* than paper cash (which offers none), surveillance resistance *stronger* than bank money (which offers none), and a default the citizen controls with their own hands. Cash with a carbon-copy receipt, where the only copies belong to the people who made the payment.

The system comes in three flavours, fixed by who is named when:

- **A1 – addressed cheque, public payer.** The note names its recipient cryptographically (only Bob's identity can deposit it; a thief who photographs the paper gains nothing) and its

issuer openly (payroll, invoices, audited disbursements).

- **A2 – addressed cheque, private payer.** As A1, but Alice's identity is hidden from everyone *except Bob*; the flavour for paying privately while still handing the recipient a receipt that names you. This is the cryptographically expensive flavour; Sections 9–10 locate the cost and explain it.
- **B1 – bearer cheque, public payer.** Whoever holds the opening can deposit it (walking-around money; the paper surface), and the depositor's identity is delivered (encrypted, bilaterally) to the issuer. The fourth combination, a bearer note from a *hidden* issuer, is not merely undesirable; it is impossible (Theorem 8).

3 Related Work

Every predecessor gives up one of the two goods. The anonymity line gives up accountability; the auditable line gives up doing without an opener. No prior system provides a *mandatory, bilateral, collusion-resistant* receipt with *no system-wide opener*:

System	Unlinkable	Mandatory receipt	Collusion-resistant	Trusted opener?	Bearer/paper
Chaum e-cash [2]	issuer-blind	no	no	central bank	no
Zerocash / Zcash [3]	yes	no (opt. viewkey)	no	no	no
Tornado / Privacy Pools [5]	yes	no (innocence)	no	no	no
PRCash [6]	yes	regulator-facing	via auditor	**yes** (auditor)	no
UTT [7]	yes	budget-gated	via committee	**yes** (committee)	no
PEReDi [8]	yes	regulator-facing	via threshold	**yes** (maintainers)	no
Platypus [9]	yes	regulator-facing	via central bank	**yes** (cen. bank)	no
CBDC (e-CNY, digital euro) [1]	no	n/a	n/a	issuer sees all	no
BUCK Notes (A1, A2, B1)	**yes**	**yes**	**yes**	**no**	**yes**

The auditable-privacy line (PRCash, UTT, PEReDi, Platypus) is the closest comparison. All four restore accountability to unlinkable payments, and all four do it by appointing an authority (an auditor, a committee, the central bank) with the power to de-anonymise. That authority is exactly what a breached, purchased, or repurposed institution turns into a population-scale surveillance instrument. BUCK's accountability is *counterparty-held*. The only institutional anchor is the per-identity KYC issuer, which (Theorem 3) can map a *surrendered* identity to a person but can never produce the link itself: compulsion runs depth-first, through named individuals – never breadth-first, across the system.

On the anonymity line, Zerocash-style view keys are *optional* and payer-controlled – precisely not mandatory – and the innocence-proving of Privacy Pools is voluntary exculpation, not compellable evidence. On the bearer axis, prior physical-form digital cash (Casascius coins, BIP-38 paper wallets) wraps a bearer secret with no identity semantics at all. BUCK's addressed flavours make the paper artifact itself useless to anyone but the named recipient.

4 Model, Adversaries, and the Invariant

The terms, in brief: a person's *Identity* is a single curve point derived from their KYC record; *accounts* are disposable key pairs, each carrying an encrypted copy of its owner's Identity and a credential that proves certification without identifying the certificate; and a *registry* accumulates the set of certified Identities so that membership in it can later be proven in zero knowledge.

Formally: an Identity is a KYC-certified point $M = m\,G \in \mathbb{G}_1$ (BN254), where the scalar $m = H(\mathrm{id})$ is a preimage-resistant hash of the certified attribute record id. A person registers any number of on-chain accounts. Each account X stores its owner's Identity ElGamal-encrypted under its own key (ElGamal: the standard public-key scheme whose ciphertexts can be freshly re-randomised) as $E_{\mathrm{addr}}[X] = (R_X, C_X) = (r_X G,\ M_X + r_X\, pk_X)$ with $pk_X = sk_X G$. Alongside it sits a Pointcheval-Sanders (PS) credential: the issuer's signature over m, re-randomised by the client so that not even the issuer can recognise it, proving the Identity was certified without revealing which certificate. The registry also maintains an incremental Merkle accumulator over identity points, yielding one compact root $\mathsf{rt}_{\mathsf{id}}$ (Poseidon leaves) standing for the whole set, membership provable against it – updated commit-before-use as accounts register.

These two secrets carry different authority. The identity scalar m grants *read* capability (decrypt ciphertexts addressed to $M = mG$, verify receipts, name a counterparty) and is disclosed to every counterparty by design: Bob learns Alice's m when she pays him, and vice versa. The PS credential σ, in contrast, grants *write* capability (register accounts, authorize spends) and is held only by its owner. The system's security rests on the fact that m alone, without σ, is insufficient for any mutating operation: an adversary who knows m can name the Identity but cannot impersonate it. A *payment* moves value v from payer P to recipient D.

4.1 Adversaries and privacy goals

The model's three properties: no one can reconstruct who paid whom from the chain; nothing on chain reveals either party's Identity to outsiders; and the two parties can always identify each other. The first two must hold against the strongest adversary available: the issuer itself, which knows every identity it ever certified.

Definition 1. *Three properties, held against adversaries of increasing power:*

- ***(Bulk-tracing resistance.)*** Mallory*, a passive observer with full chain history and unbounded compute but no party's secrets, cannot reconstruct who paid whom, how much, or link a mint to its spend(s).*
- ***(Third-party non-disclosure.)*** *Nothing on chain reveals either party's Identity point M to a non-participant, even in aggregate across many payments (*including to the issuer*, below).*
- ***(Mutual decryptability / accountability.)*** *Each of the two participants, from data they hold plus public chain state, can produce a sound receipt naming the other's registered M; and a* colluding *pair cannot arrange an accepted payment for which no such receipt exists.*

The first two properties must hold against the strongest realistic deanonymiser, the ***Issuer****: the KYC trust anchor that holds, for every client it certified, the plaintext identity* $(\mathrm{id}_i, m_i, M_i)$ *and the credential σ_i it issued, strictly more than Mallory knows. We treat a* compelled or wholly breached *issuer in Sec. 4.3.*

4.2 The non-deniable-receipt invariant

One definition carries the accountability half of the paper. Its first two clauses are standard: receipts can always be produced, and never forged. The third is the new one: even a payer and payee working together cannot make an accepted payment that leaves no receipt.

Definition 2. (Non-deniable receipt.) *A protocol satisfies the* ***non-deniable-receipt invariant*** *if for every accepted payment there exist a receipt $\mathcal{R}$ and an efficient public predicate* RcptVerify *such that:*

- ***(Completeness.)*** *The recipient D can compute $\mathcal{R}$ from data they hold plus public chain state, and* $\mathsf{RcptVerify}(\mathcal{R}) = 1$ *names a registered Identity M_P and value v.*
- ***(Soundness / non-frameability.)*** *No PPT adversary produces $\mathcal{R}'$ with* $\mathsf{RcptVerify}(\mathcal{R}') = 1$ *naming an Identity that did not author the payment.*
- ***(No deniability.)*** *No PPT payer can cause an accepted payment to D for which* no *accepting receipt naming the payer exists – even in collusion with D.*

Completeness alone protects a self-interested recipient; the third clause is what forbids a *willing* pair. Symmetric statements bind the recipient's identity toward the payer (Theorem 11 gives the per-flavour content).

4.3 Issuer-blindness: accountability without a bulk opener

The KYC issuer is the system's only trust anchor, and the natural objection to any *accountable privacy* claim is that the issuer is a bulk-deanonymisation backdoor. It is not. Issuance is *address-blind*: the issuer signs the identity scalar $m = H(\mathrm{id})$ once (a PS credential over m alone) and is thereafter done. The client derives each on-chain account *offline* (a fresh key pair (sk, pk), a self-formed ElGamal credential $E_{\mathrm{addr}} = (rG,\ M + r\,pk)$, and a *rerandomised* credential σ'), without the issuer ever seeing pk. The issuer's view is therefore, per certified client i, the tuple $(\mathrm{id}_i, m_i, M_i, \sigma_i)$ (the plaintext identity) plus public chain state. It holds *no* chain addresses.

Theorem 3. (Issuer-blindness.) *Under statistical unlinkability of PS rerandomisation and IND-CPA security of ElGamal on $\mathbb{G}_1$ (equivalently DDH), an issuer (even one compelled or wholly breached, in possession of $\{(\mathrm{id}_i, m_i, M_i, \sigma_i)\}_i$ and the entire chain) has negligible advantage in matching any issued identity M_i to its on-chain registration (pk_j, E_j, σ'_j). Compulsion or breach yields the* set *of certified identities but no* map *to chain accounts; deanonymisation remains per-account, requiring that account's sk or a counterparty receipt.*

Proof. Two locks. *Credential lock.* σ'_j is a rerandomisation of some issued σ_i; by statistical unlinkability the two carry zero mutual information, so even an unbounded issuer cannot recognise its own signature. *Plaintext lock.* The issuer *knows* M_i, so its task is not decryption but *plaintext-checking*: does $E_j = (R, C)$ encrypt M_i? That is deciding whether $(G, pk_j, R, C - M_i)$ is a DDH tuple, and an algorithm answering it for a chosen M_i breaks IND-CPA (submit $m_0 = M_i$, m_1 random). Knowing the plaintext thus confers no advantage. With both locks shut and no stored pk, the issuer is reduced to an independent DDH decision per (identity, account) pair. □

This is what *no trusted opener* means here: the closest thing to an authority cannot open transactions in bulk, even against itself. Under compulsion it can only *verify* a link some party surrenders – map a receipt-disclosed M to the real person – never *produce* one. Two caveats bound the theorem:

- **Computational, not information-theoretic.** The plaintext lock is DDH; only the credential lock is statistical. A discrete-log break – in particular a quantum computer running Shor against $\mathbb{G}_1$ – turns the harvested $\{M_i\}$ plus the archived chain into a wholesale, retroactive deanonymisation of *past* identity-bound traffic. The breached or compelled issuer's KYC records are precisely that harvest corpus. Migration of the identity substrate ahead of cryptanalytic risk is a deployment obligation, not an afterthought (Sec. 11).

- **Retroactive, not prospective.** Address-blind issuance protects *past* registrations unconditionally against *future* issuer compromise; but a coerced issuer can change the protocol going forward (demand *pk* at KYC, record addresses, backdoor credentials). Issuer-blindness is a statement about issuances already performed under the honest protocol.

4.4 The compulsion calculus

The invariant and issuer-blindness, composed, induce a precise answer to the question *who can a court reach, and what happens when they refuse?* We state it as three properties. Call a party's *opening material* for a payment the secrets from which an accepting receipt can be computed: the note opening (its randomness ρ and identity payload) together with, per flavour, the holder's identity scalar or account key needed to decrypt the counterparty ciphertext.

Proposition 4. (Existence and exclusivity.) *For every accepted payment: (i) accepting receipts naming each counterparty exist and are computable by the other counterparty from material they hold (Def. 2, clauses 1 and 3; Theorem 11); and (ii) no PPT party* outside *the payment's direct participants – including a coalition of Mallory and the compelled issuer of Theorem 3 – can compute any accepting receipt for it, under the privacy properties of Theorem 14.*

Proposition 5. (No institutional bypass.) *The only compellable institution, the KYC issuer, can perform exactly one useful act under compulsion: given an identity point M* already disclosed *by a participant's receipt, it maps M to the certified legal person. It cannot – even fully breached – enumerate, sample, or statistically approximate the payments of any person (Theorem 3). No other system component (contracts, verifiers, provers, relayers) holds any secret at all.*

Proposition 6. (Refusal-resilience.) *If every direct participant in a payment withholds or destroys their opening material, then no PPT coalition of all remaining actors – the compelled issuer with its full KYC corpus, all chain history, all registry state – gains non-negligible advantage in identifying either party or linking the payment's mint to its spend. Destruction is irreversible: the opening material is the* only *trapdoor, and it never existed anywhere but in the participants' custody.*

Propositions 4–6 follow from the privacy theorems (Sec. 8) plus issuer-blindness. Their content is the *composition* – constitutional design more than cryptography:

- **Due process is fully served.** Evidence exists for every payment (no colluding pair can prevent it), it convicts only the guilty (non-frameability: a receipt cannot be forged against an innocent party; Corollary 13), and a court always has a *specific person* to direct an order at. Investigation proceeds the way warrants are supposed to work: from a known participant, one named counterparty at a time, each step producing sound evidence.

- **Bulk surveillance is not for sale.** There is no database to subpoena, breach, or buy. The marginal cost of surveilling one more citizen is coercing one more citizen: publicly, individually, through legal process directed at a person who can see it happening. The system makes population-scale financial monitoring not illegal but *unimplementable*.

- **The last word belongs to the citizen.** Refusal has personal legal cost (contempt, adverse inference), as it should, under legitimate process. But it is *effective*: a citizen facing an unjust order, or a population facing a judiciary that has begun criminalising beliefs, can withhold or destroy keys, and the archived ledger stays dark regardless of what any institution is later compelled to do. The privacy floor survives institutional capture because no institution was ever load-bearing.

Two honest edges. First, receipts are bilateral: your counterparty's copy is beyond your control, *by design*. A criminal cannot silence their victim's receipt by destroying their own. Refusal protects a payment only when every participant elects it; exactly the case the property is for: consensual acts the regime of the day has criminalised. Second, destroying your keys also destroys *your* evidence: the standing ability to prove what you paid and to whom. Refusal-resilience is an option with a price, chosen by its holder – not a default that degrades accountability for anyone else (Def. 2 holds for every accepted payment regardless).

5 The Identity-M Principle

The properties above are only as good as the binding that produces the receipt, and the central design decision is *what* a payment binds. The obvious answer (the counterparty's account public key, the thing that actually signs transactions) is wrong. The system's history includes the scar: an early design bound addressed notes to the recipient's mint-time key pair, so losing that one key made the note permanently unspendable, and any key rotation or multi-device custody broke addressing. Worse, Sec. 9 shows the cheap forms of key binding are not merely brittle but *vacuous*: an ElGamal ciphertext does not commit to its key, so "encrypted under the same key" constraints are absorbed by witness values the prover may choose freely (Lem. 15).

The correct authority axis is the **identity point** $M = H(\mathrm{id})\,G$:

- **Identities are scarce and non-grindable; keys are free.** Anyone can mint key pairs at will, so key possession proves nothing about *who*. A registered M exists only by KYC certification, and its preimage structure denies an adversary any choice of its value:

  Definition 7. (Non-grindable identities.) *Each registered identity point is* $M = H(\mathrm{id}) \cdot G$, *where* id *is the KYC-verified attribute record and* H *(keccak256 over a canonical encoding) is preimage-resistant, modelled as a random oracle. No PPT adversary registers an identity point of its choosing; in particular it cannot register an account whose* M *equals a prescribed target* $M' + \delta G$ *for an adversarially chosen* $\delta \neq 0$.

- **Accounts are envelopes; the identity is the addressee.** A note addressed to M_{rec} is spendable from *any* account whose registration binds it to m_{rec} – including one created after the note was minted. Key loss is recovered by registering a fresh account to the same identity; no custodian, no social recovery, no issuer involvement. Dually, the receipt names the *person* (their certified M), not a disposable key.

- **Identity binding survives the time gap.** The deep obstruction in doubly-anonymous payments (Sec. 9) is that the payer's anchor exists only at mint and the recipient's only at spend; no single on-chain moment sees both. Account keys cannot bridge the gap (the recipient's spending key may not exist yet at mint). The identity point can: it is stable across the gap, and – the technical heart of the system – *everything a note must say about its parties can be said as an ElGamal re-encryption statement about identity points.*

The last observation collapses the construction into **one gadget**:

> **The gadget.** Prove, in zero knowledge, that a ciphertext is an encryption (or re-encryption) of a *registered* Identity under a target identity point – where "registered" is certified by Merkle membership of the underlying point in the registry accumulator $\mathsf{rt}_{\mathsf{id}}$, bound to the same blinded commitment $P = M + bH$ the accompanying sigma protocol opens.

Three instantiations of the gadget – differing only in *which* identity is encrypted under *whose* point, and at which moment – produce the three realisable flavours (Table in Sec. 6). The same inversion gives the spend path its shape: at deposit, the depositor proves *their account is bound to the identity the note addresses.* That is a statement about m, checked by a sigma protocol (a classical three-move proof of knowledge, a few curve operations to verify) against the account's registered ciphertext, composed with a set-membership proof that the named point is real, and a SNARK tie

(a succinct zero-knowledge proof, constant cost to verify) that the whole gate is about *this* note. Authorization keys on the identity; the account merely carries it. And because the identity scalar m conveys no write authority – only the PS credential σ can authorize account registration or spend – m is safe to disclose to every counterparty: the receipt names a person, not a capability.

6 Construction

The construction in one paragraph: a note is a hidden commitment in a shared pool. Minting proves the pool received the money and that the note names its issuer. Spending proves – without revealing which note – that some note is yours, that your account is bound to the identity it addresses, that this identity is registered, and that all of these statements are about the *same* note.

The starting point is the system's ordinary transfer: a synchronous handshake in which each party re-encrypts their credential under the other's key and proves (Chaum-Pedersen) the re-encryption carries the same M. Both identity anchors are co-present in one call; mutual decryptability is free. A Note reorganises the same handshake through a commitment pool, splitting it across time so that mint and spend are unlinkable – and the rest of the construction is the work of carrying the bilateral binding across that split.

6.1 The pool, the commitment, the nullifier

The pool is a Tornado-style Merkle tree of note commitments, hashed with Poseidon (a hash designed to be cheap inside proof circuits), keeping a rolling window of recent roots and a set of spent *nullifiers* – one-way serial numbers, revealed only at spend, that make a double-spend detectable without linking the spend to its note. A note is the opening tuple; its on-chain commitment is

$$cm = \text{Poseidon}_5(\text{flavor},\ v,\ \rho,\ idHash,\ \text{predicate}),$$

where ρ is the note's secret randomness and $idHash$ is a Poseidon hash of the flavour's *identity payload* – the commitment to everything the note says about its parties. Batch mints are proven by a Groth16 SNARK (proofs a few hundred bytes, constant gas to verify; one circuit per pinned batch size N) that opens every commitment, range-checks the values, sums them to the public escrowed total, and folds the leaves into the attested new root. The contract escrows exactly the proven total in the same atomic call.

Spends are proven by a single flavour-agnostic circuit: the prover shows knowledge of an opening of *some* tree leaf under a recent root, with public (root, nf, face, recipient, chainid), where the nullifier

$$nf = \text{Poseidon}_3(\rho,\ idHash,\ 4242)$$

is one derivation for every flavour (the constant keeps nullifier preimages distinct from commitment preimages). Double-spends are rejected by set membership. No account key enters the hash: spend *authorization* is layered per flavour on top of the nullifier. That is what makes key-loss recovery and account-agnostic deposit possible (Sec. 5) – and what creates the binding obligation the rest of this section meets.

6.2 Three flavours, one gadget

What $idHash$ commits, and which direction of the gadget runs at which moment, defines the flavour:

Flavour	idHash commits	Issuer named…	Recipient named…
A1	$(eNote,\ m_{\text{iss}},\ \sigma_{\text{Schnorr}})$	at mint, in the clear (batch Schnorr)	at spend: coupling to M_{rec} via $eRec$
A2	$(eNote,\ eIss)$	at spend, to recipient alone: decrypt $eIss$	at spend: same coupling, target M_{rec}
B1	$(m_{\text{iss}},\ \sigma_{\text{Schnorr}})$	at mint, in the clear	at spend: depositor binding $eDepForIss$ to issuer

with the per-flavour ciphertexts (all ElGamal over identity points):

- **A2 (addressed, private issuer).** The issuer encrypts *its own* registered identity under the recipient's identity point: $eIss = (r'G,\ M_I + r'M_{\text{rec}})$, decryptable by m_{rec} alone; the note value rides in $eNote = (r_nG,\ vG + r_nM_{\text{rec}})$. At mint, a blinded re-encryption sigma (`verifyIssuerReenc`) proves $eIss$ re-encrypts the *minting account's registered* identity – the anti-framing anchor – without revealing it. Each leaf's $eIss$ is also a public input of the A2 mint SNARK (the *leaf-tie*): no leaf can lack a binding, and no binding can float to another leaf.

- **A1 (addressed, public issuer).** The issuer is named openly (a Schnorr signature over the batch under their registered key, checked at mint), and the recipient's identity is encrypted *under itself*: $eRec = (r'G,\ M_{\text{rec}} + r'M_{\text{rec}})$, so exactly the addressed identity can open and answer for it.

- **B1 (bearer, public issuer).** No recipient exists at mint (Theorem 8 makes this the *only* coherent bearer cell); at spend the depositor encrypts their own identity under the public issuer's key, $eDepForIss = (rG,\ M_{\text{dep}} + r\,pk_{\text{iss}})$, and proves it carries the same identity their payout account is bound to.

6.3 The spend-side composition

Every addressed deposit (`spendCoupledA1`, `spendCoupledA2`) verifies, in one transaction, the flavour-agnostic note proof plus three mutually-bound gates:

1. **The deposit-coupling sigma** (`verifyDepositCoupling`; EIP-196, Ethereum's native curve arithmetic): the depositing account `msg.sender` is registered and bound to a scalar m, and the supplied ciphertext $eEnc$ decrypts under m to the point blinded inside the commitment $P_I = M + bH$ – a perfectly-hiding Pedersen-style point published by the sigma.

2. **The bound membership proof** (`identity_membership_g1tie`, Groth16): P_I's underlying point is a member of the registry accumulator $\mathsf{rt_{id}}$. *Bound* does the work: the verifier derives its public inputs from the same P_I the sigma committed, so the two gates cannot be answered with different points.

3. **The note ↔ eEnc tie** (`note_binding` / `note_binding_a1`, Groth16): the ciphertext the first two gates verified is keyed to the identity material committed in the $idHash$ behind *this spend's nullifier* – the gate is about this note, not a depositor-substituted ciphertext. The tie shares the nullifier with the spend proof and P_I with the sigma.

```
// inside Notes._spendCoupled (abridged; A2 passes a1Layout=false, A1 true):
require(spendVerifier.verifySpend(proof, root, nullifier, face, recipient, chainid));
require(reg.verifyDepositCoupling(msg.sender, eEnc, dc), "bad deposit coupling");
//                                                 dc.P_I --- the same point ---v
_verifyIdentityMembership(membershipProof, dc.P_I.X, dc.P_I.Y);
_verifyNoteBinding(noteBindingProof, nullifier, a1Layout, face,
                   eEnc, dc.P_I.X, dc.P_I.Y);
```

The composition discipline – every proof's public inputs derived on chain from the same objects (`msg.sender`, $eEnc$, P_I, the nullifier, the face), never taken from the prover's bytes – is what makes the gates impossible to decouple under collusion. We commend the pattern to designers beyond this system. For B1 the depositor-binding sigma (an Okamoto-style proof coupling the payout account, $eDepForIss$, and $P_{\text{dep}} = M_{\text{dep}} + bH$) replaces gate 1, the same bound membership is gate 2, and no tie is needed (the depositor names *itself*; there is nothing to substitute).

For A2, $eEnc$ is a fresh *re-encryption* of the committed $eIss$ (same plaintext, new randomness): re-using the mint-time ciphertext – a public input of `Notes.mint` – would link spend to mint and destroy the flavour's anonymity. Hence the tie proves re-encryption *equivalence*, never equality. For A1, the committed payload has no second ciphertext, so the tie runs through the note's own value ciphertext with the face public; the next subsection explains why that public face is not incidental but the soundness linchpin.

6.4 The A1 tie and the face-pinning principle

ElGamal ciphertexts are not key-committing: given the randomness r_n (which travels with the opening), the "key" of $eNote = (r_nG,\ C_n)$ can be re-explained as any m' by absorbing the difference into a free plaintext, $V' = C_n - (r_nm')G$. A binding circuit that witnesses the plaintext freely therefore proves nothing about the addressed identity. The A1 tie pins the plaintext *publicly*: its statement fixes $eNote.C = (v + r_n m_{\text{rec}})G$ with v a public input that the contract sets to the spend's `face` – itself bound to the note's committed value by the spend SNARK over the same nullifier. With v and $r_n = \log_G eNote.R$ determined,

$$m_{\text{rec}} = (\log_G eNote.C - v) \cdot r_n^{-1}$$

is *unique*: only the identity the note was addressed to can satisfy the relation (Theorem 10, A1 clause; executable witness in Appendix 12). The observation generalises: when a binding must travel through a non-committing encryption, *publishing the plaintext's already-public component is the cheapest commitment.*

6.5 The receipt, assembled

A receipt is a four-link proof chain, every link checkable by a third party from public state plus the holder's disclosed material: (a) *opening* – the disclosed tuple re-hashes to a commitment; (b) *minted* – that commitment is a leaf under a historical root; (c) *issuer* – the flavour's issuer binding (Schnorr in the clear, or the decrypted-and-membership-certified $eIss$); (d) *paid* – the nullifier consumed and face transferred. The recipient-naming direction is the spend-side composition above. Link (c) is where the invariant lives or dies; Sec. 9 is about keeping it alive when the issuer is anonymous.

Because the material behind the chain is held by *both* parties – the issuer created the identity payload at mint, the recipient was handed it with the note, and the spend event publishes the rest – either party can assemble the receipt: the depositor's copy and the issuer's "I paid X" copy differ only in who proves their own account-to-identity tie. Once the parties' identity preimages are disclosed *in* the receipt, every note-side check is deterministic (re-hash the payload, decrypt the addressed ciphertexts), which is what makes the two copies verify identically. The end-user serialization of this chain – a printable, re-scannable slip – is Sec. 10.2.

7 The Forced Taxonomy

Who can be named, and how, is fixed by what is known at mint time – the axis that defines the flavours. One of the four nominal cells (bearer × private issuer, "B2") is not merely undesirable but impossible.

Theorem 8. (No private-issuer bearer note.) *No bearer note can both (i) deliver its issuer's identity to its eventual redeemer (mutual decryptability) and (ii) withhold its issuer's identity from non-redeeming holders and observers (issuer privacy).*

Proof. The argument is information-theoretic. A bearer note's redeemer is unknown at mint: it is whoever comes to hold the opening, which circulates among holders by construction. For the redeemer to name the issuer, the note must carry an issuer-identity binding that the redeemer can open; the issuer creates that binding at mint, when the redeemer is unknown, so it must be openable under key material the eventual redeemer will hold. Exactly two sources of such material exist. (1) *The redeemer's own registered secret*: but the issuer cannot target a key or identity point it has never seen, and "whichever member of the public eventually holds the paper" names no point – targeting one would require predicting the redeemer, i.e. the note would be addressed, not bearer. (2) *Note-bound material derived from the opening itself* (e.g. a key derived from ρ): then *every* holder along the chain of custody can open the binding, so the issuer's identity is disclosed to all holders – and a bearer instrument's holders are, in the limit, the public – violating (ii). A binding openable by neither is openable by the redeemer in particular, violating (i). Hence a hidden issuer requires a designated recipient: bearer and private-issuer are mutually exclusive. □

The contrast with A2 is exact: A2 reveals the issuer to *one chosen party* by encrypting under a point only that party's identity scalar opens – which is possible precisely because the recipient is designated. The realisable set is therefore $\{A1, A2, B1\}$, and the implementation enforces the boundary structurally: the mint circuit's per-leaf issuer-mode signal admits no private-issuer bearer codepoint, so a B2 leaf is not merely rejected but *unprovable*.

8 Security

Five results carry the system: the deposit gate is sound (a depositor really is bound to the identity they claim); the tie is sound (the claim is about the very note being spent); every payment yields receipts in both directions; no coalition can avoid them; and nothing else leaks. The only available cheat is self-sabotage: a coalition can un-name its own note – making it unspendable – never frame an innocent. Full reductions are Theorems 1–12 of the companion *Proofs* document; each result here carries an executable witness (Appendix 12). Assumptions: discrete log / DDH on BN254 $\mathbb{G}_1$; Poseidon collision-resistance and PRF behaviour (ROM); Groth16 knowledge-soundness and zero-knowledge; Schnorr/Okamoto special soundness; an independent NUMS point H (no known $\log_G H$); and Assumption 7.

Theorem 9. (Deposit gate soundness.) *If a coupled spend is accepted, then except with negligible probability there exist extractable (m, sk_{dep}, b) such that: the depositing account is registered with credential binding it to identity scalar m; the supplied eEnc decrypts under m to the point M^* committed in $P = M^* + bH$; and M^* is a member of the registry accumulator $\mathsf{rt_{id}}$. The same P satisfies all three clauses: answering the sigma with one point and the membership with another requires $\log_G H$.*

Theorem 10. (Note $\leftrightarrow$ eEnc tie.) *If the binding proof is accepted for nullifier nf, then except with negligible probability the extractor produces an opening $(\rho, idHash, \ldots)$ with $nf = \text{Poseidon}_3(\rho, idHash, 4242)$ – the very nullifier the spend consumed – and:*

- ***(A2 layout.)*** *$idHash = \text{Poseidon}_8(eNote, eIss_0)$, the supplied eEnc is a re-randomisation of the committed $eIss_0$ under $M_{\text{rec}} = m_{\text{rec}}G$, the committed ciphertext decrypts under m_{rec} to M_I, and $P_I = M_I + bH$ is the sigma's committed point.*
- ***(A1 layout.)*** *$idHash = \text{Poseidon}_8(eNote, m_{\text{iss}}, \sigma)$, $eNote = (r_nG,\ vG + r_nM_{\text{rec}})$ with v the spend's public face, eEnc is a fresh encryption of M_{rec} under itself, and $P_I = M_{\text{rec}} + bH$. The public v makes m_{rec} unique (Sec. 6); a holder of a stolen opening with identity $m' \neq m_{\text{rec}}$ has no satisfying witness.*

Composed with Theorem 9 over the shared P_I, the depositor's authenticated identity is *the identity the spent note addressed, and (A2) the membership-certified point* is *the committed ciphertext's decryption.*

Theorem 11. (Mutual decryptability, all flavours.) *For every accepted payment, each party can produce an accepting receipt naming the other's registered identity: A1/B1 recipients name the issuer from the in-clear M_{iss} plus batch Schnorr; A2 recipients decrypt $eIss$ under m_{rec}, with the mint binding and membership certifying it is the registered minter; A1/A2 issuers hold the addressed M_{rec} from mint; B1 issuers decrypt $eDepForIss$. All receipts are third-party checkable.*

Theorem 12. (No deniability under collusion.) *No PPT payer/recipient coalition produces an accepted payment with no accepting receipt. At mint, every leaf must carry a verified issuer binding (Schnorr or blinded re-encryption + leaf-tie; the circuit's issuer-mode signal leaves no unbound codepoint). At spend, the deposit gate (Thms. 9, 10) reverts unless the verified ciphertext – provably the* note's own *– decrypts under the depositor's authenticated identity to a registry member. A coalition can mint a note whose committed identity material is garbage, but such a note admits no membership witness and is* unspendable*: un-nameable implies un-spendable.*

Corollary 13. (Deny, never frame.) *The only deviation available to a coalition is to render a note un-nameable – in which case it cannot be spent – never to make a receipt name an innocent third party: every binding is proven over* `msg.sender`*'s* own *registered credential (mint) or the depositor's* own *account binding (spend), and Assumption 7 denies the coalition any registered identity at an offset of its choosing. Receipts are therefore evidence-grade: an accepting receipt names a party that actually acted.*

Theorem 14. (Privacy.) *Against Mallory and against the (even breached) issuer: commitments and nullifiers reveal nothing (Poseidon hiding/PRF); a spend links to its mint only through nf, whose preimage includes the secret ρ; the spend-side sigmas are HVZK and their only identity-derived public value is the perfectly-hiding $P = M + bH$; the membership and tie proofs are Groth16 zero-knowledge with public inputs derived from already-public values; eEnc is a uniform re-randomisation. The residual public surface of a spend is exactly* (face, recipient account) *– mitigated operationally by fixed denominations and relayed payouts – and of a mint, the batch total.*

Proof sketches: Theorem 9 is Okamoto special soundness for the sigma plus Groth16 knowledge-soundness for the membership, glued by the on-chain derivation of both proofs' public inputs from one P (any discrepancy yields $\log_G H$). Theorem 10 is Groth16 knowledge-soundness over the two binding relations; the A1 uniqueness clause is the face-pinning argument. Theorems 11–14 and the corollary compose these with the mint bindings and the identity-layer results (PS unlinkability, ElGamal IND-CPA, Theorem 3). Full statements: *Proofs* Theorems 7–12.

9 The Cost of Anonymity

The public-issuer flavours bind the issuer with a signature and the depositor with sigma protocols (cheap EIP-196 arithmetic, no SNARK over identities beyond the shared membership proof). What makes the doubly-anonymous flavour (A2: issuer named only at mint, recipient only at spend, the two unlinkable) categorically harder? In short: when nobody is named in the clear at the binding moment, the cheap bindings prove nothing, and a zero-knowledge proof of registry membership becomes unavoidable. This section locates that boundary: the diagnosis, the vacuity of the obvious fix, the necessity result, and the shipped architecture as the branch the necessity result permits.

9.1 The EOA-transfer mirror: the cost *is* the anonymity

A direct transfer between registered accounts gets collusion-resistance free: the mandatory approve handshake reads *both* parties' registered records in a single call – both anchors co-present, both named. A2 splits the anchors in time *by demand of its privacy goal*: the issuer's anchor is live only at mint, the recipient's only at spend, and unlinkability forbids any public value from bridging them. The A2 mint binding is, line for line, the approve handshake with one change: the recipient's identity is *blinded*. That one change is the entire gap – it surrenders the phrase "under the recipient's *registered* key" – and everything that follows is the price of buying the phrase back without unblinding anything.

9.2 The obvious fix is vacuous

Lemma 15. (Vacuity of key-equality coupling.) *Let an A2 mint constrain only that the issuer payload $eIss$ and the note ciphertext E_{note} are encrypted under the* same *key pk_{rec}, with the recipient's identity point M_{rec} a free witness. Then for any issuer-chosen pk_{bogus} the constraint is satisfiable while the note remains spendable by the intended recipient D and $eIss$ is encrypted under $pk_{\text{bogus}} \neq pk_D$.*

Proof. Set $pk_{\text{rec}} := pk_{\text{bogus}}$ and $M_{\text{rec}} := M_D + r_n(pk_D - pk_{\text{bogus}})$. Then $E_{\text{note}} = (r_nG,\ M_{\text{rec}} + r_n\, pk_{\text{rec}}) = (r_nG,\ M_D + r_n\, pk_D)$, a valid encryption of M_D under D's registered key, so the spend-side check accepts and D spends; while $eIss = (r'G,\ M_{\text{iss}} + r'\, pk_{\text{bogus}})$ is under pk_{bogus}, so a compelled sk_D decrypts $eIss$ to $M_{\text{iss}} + r'(pk_{\text{bogus}} - pk_D) \neq M_{\text{iss}}$. The free witness M_{rec} absorbs the constraint. (Executable witness: Appendix 12.) □

The lemma is the formal autopsy of an entire class of cheap designs, and the second motivation (after key loss) for the Identity-M inversion: binding to *a* key is worthless, because keys are free and ElGamal does not commit to them. The binding must reach a *registered* identity ; the issuer, holding no recipient secret, can assert registration only as a set-membership statement.

9.3 Necessity: set-membership, at one of exactly two moments

Proposition 16. (Conditional necessity.) *Let* Π *be a protocol that, in the A2 setting, (i) preserves bulk-tracing resistance and third-party non-disclosure (issuer named only at mint, recipient only at spend, mint and spend unlinkable), (ii) closes the no-deniability clause, and (iii) achieves the binding by referencing a registered party at a* single *protocol step. Then* Π *performs a zero-knowledge proof of membership in the identity registry – over the recipient set at mint, or the issuer set at spend.*

Proof. By Lemma 15, (ii) forces Π to certify that $eIss$ decrypts, under the addressed recipient's registered identity material, to a registered *issuer* identity – binding to some key alone does not

close the clause. This certificate references a registered party: the recipient (at mint), or the issuer that $eIss$ re-encrypts (at spend). By (iii) it is a single step's reference, either in the clear or in zero knowledge. *In the clear is ruled out:* revealing the referenced identity or key on chain names a party and makes them linkable across mint and spend, violating (i); importing the spend-side ciphertext (or a deterministic commitment to it) into the mint to bind there re-links mint and spend, also violating (i). *Zero-knowledge is set-membership:* a reference certifying the referenced party is registered without revealing which is, by definition, a zero-knowledge proof of membership in the registry. As (i) excludes the in-clear case, Π performs this. □

9.4 The shipped system is the predicted branch

Proposition 16 permits exactly two homes for the membership proof, and the system's own history visited both. An earlier design (retained in the retired drafting skeleton of this paper) placed it at *mint*: a per-leaf circuit opening the issuer's credential, proving the recipient's account-keyed credential a member of an account registry, and coupling the payload under the recipient's *account key* – at roughly 10^6 R1CS per leaf, on top of the account-pinning brittleness of Sec. 5. The shipped system takes the *spend* branch, and the Identity-M inversion is what makes the branch practical:

- **The membership moves to where its prover is.** At spend the depositor is live, holds m_{rec}, and can prove membership of the relevant point – the *issuer's* decrypted identity for A2 (closing the clause exactly as the proposition demands), the recipient's own for A1. The proof is per-*spend*, not per-minted-leaf, and proving cost lands on the interested party's own hardware; the chain only ever verifies.
- **The statement splits along its natural seam.** Account-binding and decryption-to-a-committed-point are sigma-protocol material (Theorem 9; ˜90K gas to verify); only set-membership and the note tie need SNARKs at all. The composition is re-joined by deriving every public input from the shared objects – the discipline of Sec. 6.
- **Relocation creates one new obligation, and it is dischargeable.** Binding at spend means the verified ciphertext is the *depositor's submission*, so it must be tied to the note being spent – otherwise a stolen opening redeems with a self-addressed ciphertext, and a coalition substitutes a nameable one for an un-nameable commitment. That obligation is Theorem 10's two circuits, including the A1 face-pinning trick – the genuinely new constraint engineering this relocation demanded.

The result: the doubly-anonymous flavour, predicted expensive, costs a measured ∼340K gas of extra verification per spend over the bearer flavour (Sec. 10) and seconds of off-chain proving – practical today. Whether restriction (iii) can be dropped – whether a binding amortised across steps, a two-party mint with a blinded recipient contribution, or a witness-encryption construction evades set-membership entirely – remains the paper's central open problem (Sec. 11): an unconditional impossibility would prove the SNARK necessary; a loophole would be a cheaper design.

9.5 A mechanism remark: the recipient-burden posture

Before the tie circuits shipped, the design contemplated an economic backstop: an honest A2 recipient verifies the binding *at delivery* (decrypt $eIss$, check the registered identity) before accepting. An audit regime with penalty L and probability q then makes verify-then-accept dominant whenever qL exceeds the collusion benefit – and, by Corollary 13, the residual off-equilibrium outcome is only

ever an un-nameable note held by an always-identifiable depositor, never a framed third party. With the tie enforced on chain the mechanism no longer carries weight, but we record it: delivery-time verification remains good wallet practice, and the analysis explains why even a deployment that lags the verifier rollout degrades to a bounded, recipient-borne risk rather than an open hole.

10 Implementation and Evaluation

Everything measured below is shipped, open, and reproducible from the repository: Solidity contracts (`Notes`, `IdentityRegistry`, the generated Groth16 verifiers behind thin adapters), circom circuits, the Python wallet reference, and an end-to-end Foundry suite (`NotesE2E`) that drives one full lifecycle per flavour with *every* verifier the real Groth16 artifact – real batch mint, real spend proof against the replayed tree, real deposit sigma, real bound membership, real note-binding tie.

10.1 What is shipped

Component	Realisation
Identity substrate	`IdentityRegistry`: ElGamal credentials, PS verification, incremental Poseidon identity accumulator (commit-before-use)
Batch mint	`mint_batch[_a2].circom` per pinned N; issuer Schnorr (public) / blinded re-encryption binding + leaf-tie (private)
Spend	`spend.circom` (flavour-agnostic, depth 20); nullifier `Poseidon3(rho, idHash, 4242)`
Deposit sigmas	`verifyDepositCoupling` (A1/A2), `verifyDepositorBinding` (B1); EIP-196
Bound membership	`identity_membership_g1tie.circom`; public inputs derived from the sigma's P
Note tie	`note_binding.circom` (A2, 25 public inputs), `note_binding_a1.circom` (A1, 26: face at index 1)
Spend entrypoints	`spendCoupledA1` / `A2` / `B1`
Receipts	`AB-RCPT/1` envelope + verifier + renderer; all flavours, generated by *either* Note party (Sec. 10.2)

10.2 The receipt in hand

The invariant's deliverable is not a theorem but an artifact a citizen can hold. The wallet serializes the verified proof chain as `AB-RCPT/1`: a canonical, bit-reproducible byte string, wrapped into a plain-text payload sized for an 80mm thermal printer – a literal cash-register slip whose face names both parties in the clear and whose payload re-verifies *offline* from its own re-scanned text (`albertabuck verify -`), or anchors to any node for the chain facts.

Two design points carry the civic claims. First, the slip names **persons**, not keys: it embeds each party's full `canonical_identity_data` preimage, and verification recomputes $M = \text{keccak(data)} \cdot G$ against the named identity point – so "my counterparty is who they say they are" is a hash check, not a promise, and the accounts printed beside the names are pinned by the same proofs (the issuer bindings fold the minting account into their transcripts; the spend event anchors the payout account). Second, for Notes the receipt is **bilateral by construction**: the Identity-M payload behind the leaf's `idHash` is held by both parties, so the depositor prints their copy and the issuer prints an "I paid X" copy – for B1, naming the depositor by verifiably decrypting the spend event's `eDepForIss` – and both verify identically, because once the receipt has disclosed the identity preimages every note leg is a deterministic re-check. Eight golden renders (the five kinds plus the three issuer-side Note receipts) are pinned byte-for-byte in the test suite; the recipient's copy for an A2 note – a payment whose issuer no observer, subpoena, or breached KYC issuer could name – reads:

```
              ALBERTA  BUCK
       P A Y M E N T   R E C E I P T
------------------------------------------------
  Receipt  fo6idsp2dmel
------------------------------------------------
FROM (payer) (PRIVATE IDENTITY)
 Bob Smith
Corporate Registration - Alberta, Canada
 acct 0x0b0b000000000000000000000000000000000b0b
 idpt 0x0cef52e77ea5..0ae1f916d96b
------------------------------------------------
TO (payee - you) (PRIVATE IDENTITY)
 Alice Johnson
Alberta Identity Card - Alberta, Canada
```

```
 acct 0x0a11ce0000000000000000000000000000a11ce
 idpt 0x07a0b762be40..d418d7bf5e7c
------------------------------------------------
TRANSACTION
 Type    Note spend - A2 addressed, private issuer
 Amount              0.000250  BUCK
 When        2026-05-28 20:10  UTC
 Chain 1    Block 1234599    Log 1
 Tx      0xa2a2a2a2a2a2..a2a2a2a2a2a2
 Mint    0xaaaaaaaaaaaa..aaaaaaaaaaaa
------------------------------------------------
VERIFICATION
 status @ issue: VALID
 receipt  fo6idsp2dmel
------------------------------------------------
      a record, not a bearer instrument
```

This is the compulsion calculus (Sec. 4.4) made tangible: standing, court-grade provenance for every payment a citizen was party to – against fraud, theft, or commerce conducted with their stolen credentials – produced from their own records, without anyone's permission, forgeable by no one; while the same payment remains noise to every non-party, from the chain-analytics firm to the compelled issuer. That pairing is the shift the system exists to make: the capability to *prove* moves to the individuals who transacted, and the capability to *watch* is withdrawn from everyone else. The slip itself is a record, not a bearer instrument: it moves no funds, and discloses only what its holder chooses to disclose, one payment at a time.

10.3 Costs (measured)

Full-call gas for one mint (batch of one) and one coupled spend, per flavour, from the end-to-end suite (chain id 1, Cancun):

Flavour	`mint`	`spendCoupled*`
A1 (addressed, public)	487,875	1,035,486
A2 (addressed, private)	607,639	1,029,100
B1 (bearer, public)	487,875	701,474

Isolated per-phase verification gas:

Verification phase	A1	A2	B1
note proof (`spend.circom`, shared)	227,256	227,256	227,256
bound membership (`g1tie`)	253,120	253,120	253,120
deposit sigma (EIP-196)	89,866	89,866	128,734
note tie (`note_binding[_a1]`)	377,257	368,457	–

The cost structure tells the paper's story in numbers. Both *addressed* flavours pay the tie – the price of "only M_{rec} can spend", about 330K over bearer – while issuer *anonymity* costs only the mint-side delta (~120K, the blinded re-encryption binding): A2's spend is within half a percent of A1's. Set-membership, the provably necessary ingredient, is one constant 253K Groth16 verify regardless of registry size. Prover side: the tie circuits are 2.4M (A2) and 2.86M (A1) non-linear R1CS constraints – the unit of circuit size – in fixed-base-only EC arithmetic (the A1 relation needs no witnessed point limbs at all, since every point is a known multiple of G). They prove in seconds under rapidsnark on commodity Apple-silicon hardware, witness generation via the circom C++ calculator. Distribution cost concentrates the same way: the tie *proving keys* are 1.4 GB (A2) and 1.6 GB (A1) one-time wallet downloads, while every other prover artifact (mint, spend, membership keys and witness generators) is 2–19 MB – and the trusted-setup transcript never ships to wallets at all. All verification is constant-gas on chain; no participant ever waits on another's prover.

10.4 Verification methodology and artifact

The same vectors flow through three independent implementations – the Python wallet reference (`py_ecc`-style BN254 arithmetic), the circom witness generators, and the Solidity verifiers – and the test suites assert byte-for-byte agreement: over 400 Foundry tests (including 18 on the tie verifiers alone and the 9-test all-real-verifier end-to-end lifecycle) and 350+ wallet tests (among them byte-pinned golden renders of all eight receipt kinds, from both Note parties' sides). The companion documents are executable: the walkthrough document runs every cryptographic step of all three flavours in one seeded session, and this paper's appendix witnesses re-run on each build. Honesty items: Groth16 needs a one-time trusted setup whose secret randomness must be destroyed, and ours currently uses development entropy – a production Powers-of-Tau ceremony is a deployment prerequisite. Formal verification of the circuits (beyond witness/constraint testing) is planned, not performed.

11 Discussion, Limitations, and Future Work

What remains: one open theory question (is the set-membership SNARK truly unavoidable?), a short list of limitations, and the civic argument restated with the system in hand.

Open problem (the necessity lower bound). Proposition 16 is conditional on a single-binding-step restriction. The unconditional question – *must* any A2-private, collusion-resistant protocol perform registry set-membership ZK somewhere? – is open. The escape routes the restriction excludes are precisely the interesting future designs: bindings amortised across multiple steps; two-party mints where the recipient contributes a blinded verifiable-decryption witness without exposing the note ciphertext; witness-encryption-style constructions; and accountability that does not bind $eIss$ at all (threshold or selective escrow – a strictly weaker trust model we reject for this system, since it reintroduces an opener).

Limitations.

- The spend's `face` and recipient account are public (an ERC-20 must pay someone a known amount); mitigations are operational (fixed denominations, relayed payouts), not cryptographic.
- Refusal-resilience and issuer-blindness are computational (DDH); archived ciphertexts are exposed to future cryptanalysis, and the KYC corpus is a harvest-now-decrypt-later target. A quantum-migration path for the identity substrate is future work with a deadline outside our control.
- The KYC issuer is trusted for *certification* (one person, one identity) and for honest-protocol issuance going forward (Sec. 4.3's prospective caveat); Sybil identities from a corrupt certifier defeat naming, though never framing (Cor. 13).
- Trusted setup: development ptau until a ceremony is run.
- The registry accumulator is append-only in the current deployment; identity revocation/rotation semantics are designed but not shipped.

The autonomy thesis, restated. A jurisdiction adopting this design gets per-instrument compliance *stronger* than physical cash ever offered: every payment carries court-grade evidence, held by exactly the parties a lawful investigation would approach anyway, while forgoing the one thing every CBDC design quietly includes: the standing technical capability of population-scale financial surveillance. We contend the forgone capability is the point. Evidence that must be sought person by person, under process the subject can see and contest and – at personal cost – refuse, is the cryptographic shape of due process. A ledger that stays dark when its citizens withhold their keys is the cryptographic shape of the presumption that free people's affairs are their own. The mathematics here does not decide which laws are just; it decides only that enforcing them runs through citizens, not over them. And that choice, once deployed, is not revocable by the preferences of any later administration: there is nothing left standing in the middle to compel.

12 Appendix A: Executable Theorem Witnesses

Each witness below executes, in this document's build, against the same `alberta_buck.wallet` reference implementation whose vectors drive the circom and Solidity test suites; the printed `RESULTS` blocks are regenerated on export (one seeded, deterministic session). They are sanity witnesses, not proofs: each exhibits the object whose existence (or impossibility-of-construction) the named result claims.

12.1 Setup: identities, accounts, registry

```
import random
from alberta_buck.wallet.bn254 import G1, ORDER, mul, add, neg, eq, rand_scalar
from alberta_buck.wallet.elgamal import elgamal_encrypt, elgamal_decrypt
from alberta_buck.wallet.unilateral_a2 import (
    IdentityTree, mint_unilateral_a2, make_receipt, verify_receipt,
)

_seed = random.Random(0xACC0)               # one deterministic transcript
rng   = lambda: _seed.getrandbits(256)
CHAINID = 1

def account(m):                    # a registered account bound to identity m
    sk = rand_scalar(rng); pk = mul(G1, sk)
    return dict(m=m, M=mul(G1, m), sk=sk, pk=pk,
                E=elgamal_encrypt(mul(G1, m), pk, rand_scalar(rng)))
def pt(P):
    return f"({int(P[0]) % 10**8:>8d}..., {int(P[1]) % 10**8:>8d}...)"
def dec(R, C, sk):                 # ElGamal decrypt on raw points
    return add(C, neg(mul(R, sk)))

m_iss, m_rec = rand_scalar(rng), rand_scalar(rng)
issuer, rec  = account(m_iss), account(m_rec)
M_rec = rec["M"]
tree = IdentityTree()
for a in (issuer, rec):
    tree.insert(a["M"])
print(f"issuer Identity    M_I   = {pt(issuer['M'])}")
print(f"recipient Identity M_rec = {pt(M_rec)}")
print(f"registry root            = {hex(tree.root())[:20]}...")
```



```
issuer Identity    M_I   = (24518486..., 10011201...)
recipient Identity M_rec = (48294773..., 86086204...)
registry root            = 0x21b0a34077b3d8355c...
```

12.2 Witness: Lemma 15 (key-equality coupling is vacuous)

The attack object the lemma constructs: a note spendable by D whose issuer payload is keyed to a bogus key, satisfying the naive same-key constraint via the free witness.

```
sk_D, pk_D = rec["sk"], rec["pk"]
M_D        = rec["M"]
pk_bogus   = mul(G1, rand_scalar(rng))
r_n, r_p   = rand_scalar(rng), rand_scalar(rng)

# Free witness absorbs the constraint:  M_rec' = M_D + r_n*(pk_D - pk_bogus)
M_rec_free = add(M_D, mul(add(pk_D, neg(pk_bogus)), r_n))
E_note = (mul(G1, r_n), add(M_rec_free, mul(pk_bogus, r_n)))   # "same key" pk_bogus
eIss   = (mul(G1, r_p), add(issuer["M"], mul(pk_bogus, r_p)))  # also under pk_bogus

assert eq(dec(*E_note, sk_D), M_D)           # note still decrypts for D: spendable
assert not eq(dec(*eIss, sk_D), issuer["M"])  # compelled sk_D cannot name the issuer
print("naive key-equality coupling: SATISFIED by the adversary, yet")
```

```
print(f"  E_note decrypts under sk_D to M_D : True   (note spendable by D)")
print(f"  eIss   decrypts under sk_D to M_I : False  (receipt destroyed)")
print("-> binding to 'a key' is vacuous; the binding must reach a REGISTERED identity")
```

```
naive key-equality coupling: SATISFIED by the adversary, yet
  E_note decrypts under sk_D to M_D : True   (note spendable by D)
  eIss   decrypts under sk_D to M_I : False  (receipt destroyed)
-> binding to 'a key' is vacuous; the binding must reach a REGISTERED identity
```

12.3 Witness: Theorems 11 / 12 (A2 receipt; un-nameable is un-spendable)

The honest A2 path: the recipient alone produces a receipt naming both parties. The colluding path: an $eIss$ keyed to a throwaway point decrypts to garbage that is *not* a registry member – the membership gate has no witness, so the note cannot be spent.

```
note = mint_unilateral_a2(issuer["sk"], issuer["E"], M_rec,
                          v=1000, rho=rand_scalar(rng),
                          issuer=0xA11CE, chainid=CHAINID, rng=rng)
receipt = make_receipt(m_rec, note, issuer=0xA11CE, chainid=CHAINID,
                       tree=tree, rng=rng)
res = verify_receipt(receipt, issuer["pk"], issuer["E"], tree.root(), tree)
assert res.valid and eq(res.issuer_M, issuer["M"]) and eq(res.recipient_M, M_rec)
print(f"A2 receipt: {res.reason} -- names issuer {pt(res.issuer_M)}")
print(f"            and recipient {pt(res.recipient_M)}, value {res.value}")

# Collusion: key eIss to a throwaway point.
pk_throw  = mul(G1, rand_scalar(rng))
eIss_bad  = elgamal_encrypt(issuer["M"], pk_throw, rand_scalar(rng))
M_garbage = elgamal_decrypt(eIss_bad, m_rec)
print(f"colluding eIss decrypts under m_rec to {pt(M_garbage)}")
print(f"  registered identity? {tree.contains(M_garbage)} -> membership unprovable;"
      f" deposit gate reverts: un-nameable => un-spendable")
```



```
A2 receipt: VALID -- names issuer (24518486..., 10011201...)
            and recipient (48294773..., 86086204...), value 1000
colluding eIss decrypts under m_rec to (87647643..., 90375864...)
  registered identity? False -> membership unprovable; deposit gate reverts: un-nameable => un-spendable
```

12.4 Witness: Theorem 10 (A2 layout – substitution cannot bind)

The colluders *can* build a tie witness over their bogus ciphertext (they know its randomness) – but its $idHash$ is not the spent note's, so the in-circuit nullifier disagrees with the one the spend consumed, and the on-chain public-input derivation rejects the proof.

```
from alberta_buck.wallet.note_binding import make_note_binding_witness

s, b = rand_scalar(rng), rand_scalar(rng)
w_honest = make_note_binding_witness(
    rho=note.opening.rho, eNote=note.eNote, eIssCommitted=note.eIss,
    s=s, m_rec=m_rec, b=b, M_I=note.M_I, r_iss=note.r_prime)

r_bad = rand_scalar(rng)
eIss_sub = elgamal_encrypt(issuer["M"], M_rec, r_bad)       # nameable substitute
w_sub = make_note_binding_witness(
    rho=note.opening.rho, eNote=note.eNote, eIssCommitted=eIss_sub,
    s=rand_scalar(rng), m_rec=m_rec, b=b, M_I=issuer["M"], r_iss=r_bad)

print(f"spent note's nullifier      = {hex(int(w_honest['nullifier']))[:20]}...")
print(f"substituted-tie nullifier   = {hex(int(w_sub['nullifier']))[:20]}...")
assert w_sub["nullifier"] != w_honest["nullifier"]
print("-> public-input mismatch: a substituted ciphertext cannot bind THIS spend")
```

```
spent note's nullifier      = 0x1e63df3688a6e7f638...
substituted-tie nullifier   = 0x21cbb7dfd6effd1be5...
-> public-input mismatch: a substituted ciphertext cannot bind THIS spend
```

12.5 Witness: Theorem 10 (A1 layout – the public face pins the identity)

The production A1 witness builder asserts the full relation: it builds for the addressed identity, and *refuses* for any other – even for a thief holding the complete opening including the note randomness – because the public face leaves no free plaintext to absorb the re-keying.

```
from alberta_buck.wallet.unilateral_a1 import mint_unilateral_a1
from alberta_buck.wallet.note_binding import make_note_binding_a1_witness

k = rand_scalar(rng)
sigma_R, sigma_s = mul(G1, k), (k + rand_scalar(rng) * rand_scalar(rng)) % ORDER
note_a1 = mint_unilateral_a1(M_rec, v=100, rho=rand_scalar(rng),
                             m_issuer=m_iss, sigma_R=sigma_R, sigma_s=sigma_s,
                             rng=rng)
t, b1 = rand_scalar(rng), rand_scalar(rng)
w_a1 = make_note_binding_a1_witness(
    rho=note_a1.opening.rho, eNote=note_a1.eNote, v=note_a1.opening.v,
    m_issuer=m_iss, sigma_R=sigma_R, sigma_s=sigma_s,
    r_note=note_a1.r_note, m_rec=m_rec, t=t, b=b1)
print(f"A1 tie witness for the ADDRESSED identity: built "
      f"(public face v = {w_a1['v']})")

m_thief = rand_scalar(rng)        # thief holds the WHOLE opening, incl. r_note
try:
    make_note_binding_a1_witness(
        rho=note_a1.opening.rho, eNote=note_a1.eNote, v=note_a1.opening.v,
        m_issuer=m_iss, sigma_R=sigma_R, sigma_s=sigma_s,
        r_note=note_a1.r_note, m_rec=m_thief, t=t, b=b1)
    print("thief witness: BUILT (must not happen!)")
except AssertionError as exc:
    print(f"A1 tie witness for a thief identity: REFUSED ({exc})")


A1 tie witness for the ADDRESSED identity: built (public face v = 100)
A1 tie witness for a thief identity: REFUSED (eNote.C != v*G + rn*M_rec)
```


12.6 Reproducing the full system

The on-chain halves of every claim are exercised by the repository's suites; the end-to-end lifecycle (real batch mint, real spend, real sigma, real bound membership, real tie, per flavour) reproduces with:

```
$ make nix-snark-note-binding nix-snark-note-binding-a1   # tie circuits + setups
$ make nix-snark-e2e-fixtures                             # all-real-verifier fixtures
$ nix develop --command forge test --match-contract NotesE2E -vv
[PASS] ... 9 tests: lifecycle, double-spend revert, per-phase gas, x3 flavours
```