MintID

PROTOCOL WHITEPAPER
NOVEMBER 2026
PROOF CORE NOT INDEPENDENTLY AUDITED

MINTID PROTOCOL WHITEPAPER

MintID Protocol Whitepaper

PUBLISHED BY

MintID

Developed and operated by Conectia OÜ (trading as Conectia PRO) · related parties stated in §9.3

Verified identity and accountability for the agentic era. *Zero-knowledge identity layer for agents.* · v2.0 · 2026-11-23

Contents

–	Abstract	3
01	Introduction	5
02	Parties, trust and notation	7
03	The three-party construction	15
04	Agents in the three-party construction	26
05	The identification as a parameter	33
06	The record layer	36
07	The disclosure channel	41
08	Security and privacy properties	46
09	Threat model and residuals	55
10	What is not claimed	60
11	Status	62
–	Glossary	67
–	References	69
A	Requirements and where this paper treats them	70
B	Time values of the protocol	72

ABSTRACT

Abstract

Reading guide and legal posture. This is the protocol whitepaper of MintID. It describes the three-party construction (an issuer that identifies, a holder that proves, a verifier that decides), the identification as a parameter of the issuer, the record layer the protocol reads from and the disclosure channel through which identity detail may leave the system, and it states the security and privacy properties of the construction with their assumptions and their limits. It stays above wire formats, encodings, parameters and pins. The schemes are named and the relation each proof proves is written in one notation (§2.4). The formal specifications are published with the source under the Apache-2.0 licence; the specification corpus is available on request, and where this paper and the corpus differ, the corpus governs. **The proof core has not been independently audited.** The audit of its implementation and the audit of holder self-revocation are scheduled, and their findings are to be published when each closes; the scope of the audit of the dispute ciphertext is written, and that audit is not yet commissioned. For each of the three compositions it introduces the project has written its own security argument, one of them a draft; they are not yet published, and no one outside the project has reviewed them (§3.1). The curve in production does not meet the 128-bit level the specification requires; the move to the curve that does is specified and not started, and comes before the audits (§3.1). Every privacy property in this paper is a claim about code that no third party has yet audited. Every statement about a piece that is not in production carries its status where it is made, in the status words of §1.3, and Table 13 gathers them (§11). The design for running the protocol on several nodes, the token economics, the governance of the MintID Foundation, the launch plan, and the open parameters, open decisions and risks are in the companion document, *Governance, Economics and Launch Plan* (v1.0). This paper is not the crypto-asset white paper of Regulation (EU) 2023/1114 (MiCA). **No token or crypto-asset is for sale, no price exists, and nothing in this paper is an offer.**

MintID is an open-source identity protocol that lets a software agent prove that an accountable, identity-verified person stands behind it, within a scope that person fixed, without revealing who that person is. What an agent proves of that person is the attestation its accepted issuer gave when it minted the agent (§4.2). A person verifies once with an accepted issuer and keeps the credential; the person, and every agent the person deploys, then proves only what a service asks. MintID itself issues nothing, certifies nothing and verifies nobody.

The protocol has three parties. An accepted issuer identifies a person, by an evidence route that its published policy grounds at an assurance grade, and issues an anonymous credential: one BBS signature over the person's claims and over a secret that only the holder knows and the issuer never sees. The holder proves what a verifier asks (that the grade is at least the one requested, that the credential has not expired and is not revoked, and, for an agent, that it holds a credential its issuer signed under the agent profile, which the issuer signs only for an agent it minted from a person's credential, within a signed scope and limit) in one zero-knowledge presentation bound to that verifier's challenge and valid for seconds only (§3.5). The verifier decides on its own service.

The protocol reads a small set of public facts from a shared record, the record layer: which issuers and verifiers are accepted, each issuer's current status commitment, and the one-time codes with which holders switch their own credentials off. No identification data of a holder, credential, agent record or presentation is ever written to it. MintID is a protocol. In production it runs today on a single-node chain, which Conectia OÜ (trading as Conectia PRO) operates under the mandate of the MintID Foundation; running it on several nodes is foreseen as the project evolves. The chain is MintID's record layer today.

0 holder identities, credentials or presentations written to the record layer	**30 s** lifetime of a fully bound zero-knowledge presentation, which is accepted once	**185 s** bound of an issuer's status revocation on the single-node production chain; one block on the emergency path	**1** identification backs every agent a person deploys

Within the protocol, identity detail leaves the system only through the disclosure channel, and only from the issuer. By default nothing leaves it outside the dispute cascade: every agent presentation carries an encrypted reference to the agent that the issuer and the MintID Foundation can open only together, though today one operator runs the issuer and executes the Foundation's key ceremonies, and both parts stay behind that operator; at a later clean regeneration, which follows the custodian's key ceremony, the specification places the Foundation's part with a custodian that operates no issuer, chosen by the MintID Foundation, itself a related party of the operator (§4.6, §9.3); the path from a recorded claim to the person's identity, under an arbiter's certified finding, is specified. Consented commercial disclosure and mandated identification, the two paths on which, under the specification, identity can leave by the person's own decision, are built and switched off.

Section 8 states thirteen properties of the construction, each with the party it protects, its assumptions, the test a reviewer can apply, its status and its limit; Section 9 states the threat model and every residual; Section 10 states what is not claimed.

01

Introduction

> Accountability without surveillance needs the proof of a claim and the identity of the claimant to travel apart.

1.1 Proof and identity travel together

The identity mechanisms in common use on the internet (federated login, identity checks repeated at each service, age verification by document upload, identity signals carried by payment networks) were built on the assumption that a person is at the keyboard and that knowing who the person is is the price of trusting what the person claims. A relying party that needs one fact, such as an age threshold or the existence of a verified person, receives the document that carries the fact: a name, a date of birth, a document number, an address. It keeps them, because that is the only shape in which the proof arrived. Over-collection follows from the shape of the proof, and data-protection law requires the relying party to minimise it.

The privacy-preserving alternatives that have been deployed widely (pseudonymous key pairs, self-issued identifiers, credentials whose issuer is whoever chose to sign) remove the over-collection and remove with it what a counterparty needs when something goes wrong: an identified person who answers for the key. A signature from a key that no one answers for proves possession of the key and nothing about who stands behind it. Anonymous credentials with accountable issuers and conditional de-anonymisation, the line of work Table 4 cites, address both failures for people; MintID applies that line of work to software agents, with the issuer's attestation of the person behind each agent, a signed scope, and a dispute path bound to each presentation.

Both failures have one cause: the proof of a claim and the identity of the claimant travel together, so that a relying party either receives both or receives neither. The design goal of MintID is to separate them with cryptography. A claim about a person, such as a threshold on the strength of the person's identification, the validity of the credential or the scope of what an agent may do, is proven without the person's identity; the identity stays with the party that established it and leaves it only on a path whose trigger, recipient and status are stated.

Software agents make the separation necessary. An agent acting for a person (buying, booking, calling a service, negotiating) is a party the relying party has never seen and cannot interview. The relying party needs to know that someone answers for the agent, with what assurance that someone was identified, and that the agent acts within what that person allowed. It usually needs no name: the name adds nothing to its decision, and a relying party that receives the name of every principal behind every agent becomes the register of who delegated what to whom. An identity layer for agents therefore has to prove standing, assurance and scope while revealing none of the identity, and has to keep a path open, under stated conditions, from the agent back to the person.

1.2 What the protocol does

An issuer identifies, a holder proves, a verifier decides. The identification is a parameter of the issuer: the protocol fixes what a credential says about an identification, not how the identification is done (§5). The record layer holds the public facts the three parties rely on; no presentation is ever written to it, and the verifier only reads it to decide (§6). Within the protocol, identity detail leaves the system only through the disclosure channel, and only from the issuer (§7). Section 3 describes the construction among the three parties and Section 4 the agent credential, which uses the same scheme, the same issuer and the same verifier.

1.3 How to read this paper

The status words are those of the specification corpus, as the project's roadmap applies them on the day of writing; each word has one meaning, words combine and each keeps its meaning, and every statement about a piece that is not in production carries one where it is made. *In production* means running in the production deployment, the single-node production chain and its services, and the present tense without a status word is reserved for it; *live*, running on the public testnet. *Released* means in a signed public release, as its packages or, for client software and documentation that have none, as the release's source tree; for client software, which runs in no deployment, it is the status that takes the place of in production. Server code that a release carries only in its source tree is not released: it is said by what it is, in the tree and called by no route. *Built* means merged on the main branch and, alone, in no deployment and in no release; *built and switched off*, in a deployment and disabled by its switch; *built, not issued*, its case for a credential profile: in a deployment, and issued by no issuer because the issuer's switch selects another profile; *built, not applied*, in the deployment code and live on the public testnet, not applied to the production deployment; *built on the development line*, code written on a development line that is not merged on the main branch and not released. *Specified* means that the text of record (the specification corpus, or an accepted MintID Improvement Proposal, one whose status is Frozen counting as accepted) states it, and that nothing of it is built unless the sentence says so; a rule that only a formal specification on a line not yet published states is built on the development line or decided, never specified. *Decided* means that an accepted decision record states it and that neither the text of record nor an accepted improvement proposal yet states its mechanism. A piece that is specified or decided and not in production takes effect with the release that carries it; where that release is a clean regeneration of the chain (§6.4), the paper says "at the clean regeneration that carries it". One is named: *a later clean regeneration, which follows the custodian's key ceremony* of §4.6; it is not dated. *In study* means a design or a question under study and not decided; *open*, a decision not yet taken, of which the paper states only what the text of record already states. Nothing in study or open is specified. *Scheduled* means planned for a stated stage or milestone of the sequence; *deferred*, set aside to a later stage or until a stated condition holds; *parked*, decided not to be done now, with no date; said with the record that decides it; *not built*, absent from the code and from every deployment; *not specified*, stated by no requirement of the text of record or of an accepted improvement proposal. Of an audit, *scope written* means that its scope is written and the audit is not commissioned, and *not started* that nothing of it has begun. A security argument the project wrote is *written and not reviewed*: it is the project's own, not yet published, and no one outside the project has checked it. *Unaudited*, of the proof core and of every property of §8, means that no independent audit of this implementation has closed. *Pending audit*, of a built piece, a property or a security argument, means that a named independent audit or review is still to confirm it; the paper names the audit beside it, and nothing pending audit is a finding of that audit. KYB is written "we are building KYB". The paper describes the zero-knowledge presentation rail. The classical seller rail, which uses issuer-signed attestations and is never zero-knowledge, is described in §5.1 and listed in §11.

02

Parties, trust and notation

> Three parties, one optional fourth that resolves, and a record layer that is a role, not a party.

2.1 The parties

Table 1 states, for each party, what it holds, what it learns and what it is trusted for. “Trusted for” means that a property of §8 fails if the party does not do what the row says; where nobody trusts a party, its own protection comes from conforming to the protocol.

Table 1 · The parties.

PARTY	HOLDS	LEARNS	IS TRUSTED FOR
Accepted issuer	Its signing keys, one pair per profile; its accumulator secrets; its status registry; the identification evidence, or access to it at its provider; its part of the dispute key; the accountability escrow (the issuer’s private record, never funds)	Every claim it signs except the link secret and the one-time code; at a mint, which of its principal credentials backs which agent; which network addresses fetch its public status log and when, and when a credential asks it for its status witness; which of its credentials a recorded one-time code ends; which of its agents a relying party’s pseudonym names, when the agent relays; which agent a recorded claim concerns; when a verifier that reads its published definition at each decision decides	Every value it signs; minting an agent only from a principal presentation, within its caps; drawing a fresh status reference for every signature it makes, which keeps a person’s credentials unlinkable (§2.5); removing revoked credentials (the record layer enforces only that it acknowledges self-revocations); the custody of the accountability escrow; the truth of its published policy
Holder	The credential, the link secret and the status witness; for an agent, the agent’s credential. The holder is the person, or, for an agent, the agent’s holder software, behind which the person stands	Its own claims	By no one. Its own privacy rests on the holder software it runs (the published holder builds) and on the application that embeds it (T-H)
Verifier	Its request keys and registered origins, its nonce store, its decision record and, for agents, the dispute retention record and, where it keeps one, a recognition record per agent pseudonym	The issuer, the profile, the accumulator the proof names, the result of each predicate it asked for and, for an agent, the scope values it named and the agent’s pseudonym for it	By the person, for deleting its dispute retention record after the claim window and its recognition record of a pseudonym once the longest lease an agent credential can carry (90 days under the reference mint policy) has passed since it last accepted that pseudonym, and for publishing none of its records; otherwise it protects itself by conforming: reading the record layer at each decision and consuming each nonce once

PARTY	HOLDS	LEARNS	IS TRUSTED FOR
The MintID Foundation	The protocol's authority; the admission key set; its part of the dispute key, kept in infrastructure the operator runs (under the specification, from a later clean regeneration, which follows the custodian's key ceremony, by a custodian that operates no issuer, chosen by the MintID Foundation, a related party of the operator, §4.6)	What its admissions decide; under the specification, for a recorded claim, the claim's presentation (its issuer, its predicate results, the claimant and the agent's pseudonym there), not the agent's status reference or whom it concerned	Admission under its published policy; one key record per version of its part; releasing its decryption share only as the specification says
The arbiter (the optional fourth party; specified)	No share, no status reference and no link	The claim and the evidence put to it	Deciding whether a claim is substantiated and certifying the finding on which the issuer releases identity
The record layer (a role)	The registries, the status commitments, the one-time codes and the governed values	No holder's identity; from a self-revocation, the issuer, the profile, the code and its time (Table 10)	Ordering, finality, a faithful clock, availability and authenticated reads (§6)

The MintID Foundation holds the protocol's authority: it admits issuers and verifiers under a published policy; Conectia OÜ (trading as Conectia PRO) operates the single-node production chain under its mandate. Under the admission policy a verifier is admitted only with a natural person responsible for it, identified at A2 or better by an accepted issuer; the record layer stores the issuer's commitment about that person and checks only its size (§6.4).

Every "learns" and "does not learn" in this paper assumes distinct parties. On the single-node production chain one operator stands behind the issuer, the validator, the privacy relay and the only admitted verifier, and executes the MintID Foundation's key ceremonies, and the President of the MintID Foundation is that operator's beneficial owner; §9.3 states who stands behind each role, and no property of §8 that protects against one of these parties is exercised between distinct parties there.

2.2 What sits where

Figure 1 places each party and the four data paths that connect the parties to the record layer. Three of them write. The issuer writes a signed status commitment every 30 seconds, an emergency commitment when it must end a credential's authority within one block, and its registry writes. The holder writes, through a privacy relay, a one-time code with its proof and with the credential's status reference encrypted to its issuer; the holder holds no account on the record layer and signs nothing there. The MintID Foundation's admission grants reach the record layer with each registration of a verifier and with each act of the chain's authority that changes a verifier's record. The fourth path reads: the verifier reads the record layer just before each decision, and accepts only values proven against the state commitment of a node it designates (§6.4).

Presentations never reach the record layer; identification evidence never leaves the issuer. One path does not touch the record layer and is drawn because it matters to privacy: the holder brings its status witness forward from the issuer's public status log, on a schedule common to the holders of its profile, so the issuer sees which network addresses fetch the log and when, and no request that names the credential (§3.7).

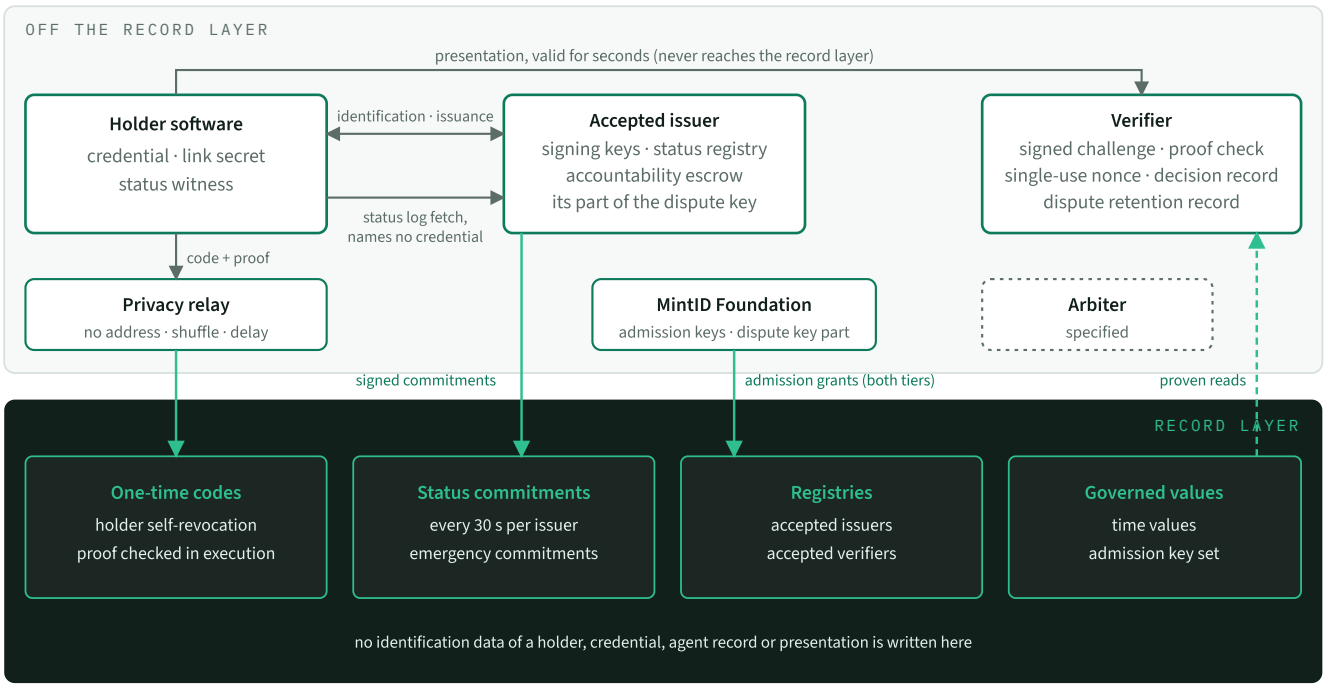

Figure 1 · Everything that identifies a person stays off the record layer. Solid arrows into the record layer write to it; the dashed arrow is the verifier's proven read from a node it designates; the holder's fetch of the issuer's public status log names no credential.

The figure shows the reference arrangement, in which each party is a distinct organisation; on the single-node production chain one operator stands behind several of these boxes (§9.3).

2.3 Invariants

The construction refuses five things. Each is a design rule from which several properties of §8 follow, and each has a limit stated where the property is stated.

Invariant 1. The record layer never learns who a holder is. No identification data of a holder, credential, credential identifier in the clear, presentation or per-holder issuance event is written to it; a self-revocation writes its one-time code and the credential's status reference encrypted to its issuer, and a verifier's record names the verifier.

Invariant 2. State grows with institutions, not people. Record-layer state grows with issuers, verifiers, validators and the protocol's own records (recorded governance decisions, self-revocations within their retention window, dispute claims, issuer-fault entries, issuers' key histories and, under the specification, disclosure receipts and registered arbiters), never with holders or presentations.

Invariant 3. Block execution is deterministic and self-contained. No network call, clock other than block time, randomness, hardware signer, remote service or foreign-function call runs in it.

Invariant 4. Backing is attested and proven; identity is not revealed. An issuer attests at mint that an agent is backed by a person's credential; the agent proves that attestation and its grade threshold, never which person. The only link from an agent to its person that the protocol keeps is in the issuer's accountability escrow.

Invariant 5. Within the protocol, identity detail leaves only through the disclosure channel, and only from the issuer. Where it is wanted for a relationship, the person decides and refusal is the default; where it is wanted for accountability, due process decides. No payment is ever the lawful basis of a disclosure. The issuer, as the party that holds the identification, its identity-verification provider and whoever holds a part of the dispute key remain subject to the law that applies to them: a court or a competent authority can require any of them to produce what it holds about a person, outside the disclosure channel, and the protocol does not prevent it.

2.4 Notation

The paper uses one notation, listed in Table 2. Groups are written additively: · is scalar multiplication, + is the group operation and 0 is the identity element. Variables are in italics. A subscript names the owner of a key (I for the issuer, F for the MintID Foundation) or the profile of a credential (id for the identity profile, ag for the agent profile); π stands for either profile.

Table 2 · Notation.

SYMBOL	MEANING	USED IN
Groups and hashes		
$G_1, G_2, G_T; \hat{e}$	The groups of the pairing-friendly curve (BLS12-381 in production, BLS24-509 as specified, §3.1), of prime order p; $\hat{e}$: $G_1 \times G_2 \to G_T$ the pairing	throughout
g_1, g_2	Fixed generators of G_1 and G_2	throughout
$\mathbb{Z}_p$	The scalars	throughout
$H_G(\cdot), H_p(\cdot)$	Hash to G_1 and hash to $\mathbb{Z}_p$ (RFC 9380), each with a domain of its own for each use	§3.3, §4.5
$h_0, \ldots, h_{\ell-1}; h_s, h_N$	The issuer's message generators for a profile; among them, those of the two blind claims	§3.2, §3.3
$g_c, h_c; h_d; h_m$	The Pedersen generators; the dispute message generator; the self-revocation message generator, all derived by H_G	§3.4, §4.6, §3.8

SYMBOL	MEANING	USED IN
Issuer		
I	The issuer's identifier, the one revealed claim	§3.2
x_π, $X_\pi = x_\pi \cdot g_2$	The issuer's BBS key pair for profile π	§3.2
V_π; a_π, $Q_\pi = a_\pi \cdot g_2$	The status accumulator of profile π; its secret and its public key	§3.7
V_ω; Q_{sc}	The scope accumulator of the scope value ω; the issuer's scope key	§4.4
ρ_j	The issuer's status commitment at epoch j: a SHA-256 tree root over its status entries and, per profile, its accumulator, the digest of its definition and its self-revocation key, per agent profile its part of the dispute key, and the accumulator of every scope value; signed with its root key and naming ρ_{j-1}	§3.7
α_j	The number of recorded one-time codes the issuer acknowledges at epoch j	§3.7
rk_I, ek_I	The issuer's root-signing key and its separately scoped emergency key (Ed25519)	§3.7
$K_{sr} = k_{sr} \cdot g_1$	The issuer's self-revocation encryption key	§3.8
$K_I = k_I \cdot g_1$	The issuer's part of the dispute key	§4.6
The MintID Foundation		
$K_F = k_F \cdot g_1$	The Foundation's part of the dispute key	§4.6
$K_d = K_I + K_F$	The dispute key	§4.6
AK_F; ak	The admission key set (Ed25519), a governed value of the record layer; one key of it	§6.4
Holder and credential		
s	The link secret, drawn by the holder, fresh per credential, signed blind	§3.3
$N = H_p(s)$	The one-time code, signed blind; the relation is computed by the holder and certified by the signature, not proven	§3.3, §3.8
y	The status reference, drawn by the issuer, fresh per credential, never revealed to a verifier; the accumulator element derived from it is published in the issuer's public status log once the credential is removed	§3.2, §3.7
γ, τ_{iss}, τ_{exp}, τ_{dob}	The grade (A1 to A4, ordered by strength), the issue instant, the expiry instant, the date of birth	§3.2
λ, u, κ, z, v	For an agent: the transaction limit, its unit, the key-custody rank, the scope reference, the principal's age floor (agent profile v2)	§4.1
$m_{id} = (y, s, I, \gamma, \tau_{iss}, \tau_{exp}, \tau_{dob}, N)$	The identity claims, with a profile-version claim	§3.2

SYMBOL	MEANING	USED IN
$m_{ag} = (y, s, l, \gamma, \tau_{iss}, \tau_{exp}, N, \lambda, u, \kappa, z)$	The agent claims, with a profile-version claim; v in profile v2	§4.1
R	The holder's commitment to its blind claims in an issuance request	§3.3
$\sigma_p, y_p; \gamma_a; s', y', z'$	At a mint: the principal's signature and status reference; the agent's backing grade; the agent's fresh link secret, status reference and scope reference	§4.2
$\sigma = (A, e)$	A BBS signature: $b = g_1 + \Sigma\, m_i \cdot h_i$ and $A = (x + e)^{-1} \cdot b$, valid if and only if $\hat{e}(A, X + e \cdot g_2) = \hat{e}(b, g_2)$	§3.2
W; W_ω	The status witness, with $\hat{e}(W, Q + y \cdot g_2) = \hat{e}(V, g_2)$; a scope witness, likewise for z and V_ω	§3.7, §4.4
Verifier and session		
v, o	The verifier's identifier and its registered origin	§3.5
Φ	The verifier's policy: a set of predicates	§3.4
t_{ref}	The reference instant, the challenge's issue instant	§3.4
n	The fresh single-use nonce	§3.5
β	The binding digest of (o, v, request key, n, digest of the signed request, finalised height)	§3.5
$\vartheta(t, n')$	The instant n' civil years before the instant t	§3.4
T_H, T_A, T_K, T_G	The heartbeat, the maximum accepted age of a commitment, its maximum future skew, the grace for acknowledging a one-time code (Appendix B)	§6.3
$h_v = H_G(v)$	The verifier's pseudonym generator	§4.5
$P_v = s \cdot h_v$	The agent's pairwise pseudonym at v	§4.5
$(C_1, C_2) = (\zeta \cdot g_1, y \cdot h_d + \zeta \cdot K_d)$	The dispute ciphertext, ζ fresh per presentation	§4.6
Self-revocation		
$(E_1, E_2) = (r \cdot g_1, y \cdot h_m + r \cdot K_{sr})$	The status reference encrypted to the issuer, r fresh	§3.8
β_{sr}	The self-revocation binding: record layer, issuer, profile, N and one recent ρ_j	§3.8
Resolution (the optional fourth party)		
$D_I = k_I \cdot C_1$, $D_F = k_F \cdot C_1$	The decryption shares, each with a proof bound to the claim and the presentation; the subscript X in K_X, k_X, D_X stands for I or F	§4.6
$M = C_2 - D_I - D_F = y \cdot h_d$	The opened value, matched by the issuer against $\{\tilde{y} \cdot h_d\}$ for every status reference $\tilde{y}$ it issued	§4.6

SYMBOL	MEANING	USED IN
φ	The arbiter's certified finding (specified)	§7.3
Proofs		
Π_{req}, Π_{id}, Π_{mint}, Π_{ag}, Π_{rel}, Π_{srp}, Π_{pop}, Π_{sh}	The issuance request, the identity presentation, the mint presentation, the agent presentation, the relay presentation, holder self-revocation, the proof of possession of a key part and the decryption-share proof	§3, §4

Each Π is a non-interactive zero-knowledge proof of knowledge in the random-oracle model (for its range components, an argument whose soundness rests on DL); "Π proves $\mathcal{R}$" means that it proves knowledge of a witness for the conjunction $\mathcal{R}$ listed with it. The self-revocation proof Π_{srp} is argued witness indistinguishable among the credentials of one issuer and profile proving against one commitment; its simulation uses the issuer's keys (§3.8). The conjuncts of one relation that share a hidden claim are linked by a shared response under one Fiat–Shamir challenge (§3.4).

What the notation leaves out. The notation names values and the relations between them. It gives no encoding, field name, domain string, byte size, bit width, generator count, schema identifier, parameter value or library version; these are in the formal specifications, published with the source, and none of them changes a property in this paper. The tuples name the claims; their order in the signature is the formal specification's.

2.5 Assumptions

Every assumption the paper uses is named once in Table 3, so that each property in §8 cites it by name. The first seven are computational assumptions on the schemes, or a model in which some of them are argued; the last eight are assumptions of trust in a party or a component, and each is a place where the construction depends on conduct rather than on mathematics.

Table 3 · Assumptions.

NAME	STATEMENT	USED BY
q-SDH	The q-strong Diffie–Hellman assumption in the pairing groups of the curve (Boneh and Boyen); on BLS12-381 its concrete level falls as one key serves more signatures (§3.1). The unforgeability of BBS signatures in the one-element, one-scalar form used here rests on it: Tessaro and Zhu prove strong unforgeability in the standard model for a uniformly random per-signature scalar, when the signer sees the messages it signs; for blind issuance their proof is in the algebraic group model (AGM), and needs that no two of the issuer's scalars collide. The library derives the scalar as the blind BBS draft does, and it refuses a status reference its key set has already signed. The derivation has not been independently analysed; it is pending audit (the implementation audit; residual 16). The soundness of VB20 accumulator membership rests on a generalised form of q-SDH, proven by its authors in the generic group model	P1, P2, P5, P8, P11, P12
DL	The discrete logarithm problem is hard in G_1 and in G_2; the binding of Pedersen commitments, and the soundness of the range proofs, of the proofs of possession and of the decryption-share proofs, rest on it in G_1. The degenerate cases of the self-revocation proof's soundness rest on it in both groups, and one of them on the q-discrete-logarithm assumption in the algebraic group model (AGM)	P1, P2, P4, P8, P11, P12
DDH in G_1 (XDH)	Decisional Diffie–Hellman is hard in G_1; the unlinkability of pairwise pseudonyms and the IND-CPA security of the dispute ciphertext and of the self-revocation ciphertext rest on it	P3, P4, P9, P10, P11

NAME	STATEMENT	USED BY
ROM	Hash-to-curve, hash-to-field and the Fiat–Shamir transform are modelled as random oracles (Pointcheval and Stern); the zero knowledge and soundness of every non-interactive proof, and the hiding of the link secret from the issuer before a self-revocation, rest on it	P1 to P5, P7 to P12
AGM	The algebraic group model, in which every group element an adversary outputs comes with its representation in the elements it received. Tessaro and Zhu's unforgeability of BBS for a signer that signs committed values without their openings, the form of blind issuance, is stated in it, with its proof in the full version of their paper, against the q-discrete-logarithm and discrete-logarithm assumptions, the first of which q-SDH implies, for scalars that do not collide; one degenerate case of the self-revocation proof's soundness is argued in it against the q-discrete-logarithm assumption. Neither is proven in the standard model in the works cited	P1, P2, P8, P11
Ed25519 (EUF-CMA)	Existential unforgeability under chosen-message attack, for status commitments, emergency commitments, the verifier's signed challenge, the Foundation's admission grants and its dispute-key record	P1, P4, P5, P7, P8, P11, P12
SHA-256	Collision resistance, for the status-commitment tree and every digest; HMAC-SHA-256 as a pseudorandom function, for the derivation of the Foundation's part of the dispute key and for the issuer's cover removals and cascade draws in its public status log	P1, P4, P5, P7, P8, P9, P11, P12
T-I (issuer)	The issuer does what Table 1 says it is trusted for	P1, P2, P4, P6, P8, P9, P11
T-H (holder software)	The holder's prover draws fresh randomness; the holder library refuses a pseudonym request for another verifier, and checks both proofs of possession of the dispute key and that the issuer's part is the one its commitment covers; the application that embeds the library supplies the trusted verifier keys and asks the person's consent as the person intends; no published holder host keeps the Foundation key record it accepted (§9.2, item 22)	P3, P4, P9
T-P (provider)	The issuer's identity-verification provider reports its checks truthfully	P1, P8
T-R (record layer)	Ordering; deterministic finality (an entry is final once recorded, within one chain: §6.4); a block time that only moves forward and does not run ahead of real time; authenticated reads; and, for availability and not for any bound, each issuer's next commitment final before its current one expires	P1, P5, P6, P8, P11, P12, P13
Clocks and cadence	Two conditions outside the record layer: the verifier's own clock, against which it judges a commitment's window and a challenge's lifetime, and the issuer building a commitment every heartbeat, on which the cascade bound counts	P6, P7
T-O (operator)	The operator, which stands behind several roles today (§9.3), does not join the records it holds in them and keeps the custodies it executes apart	P3, P4, P9, P10, P11
T-F (Foundation)	Custody of its key part, in infrastructure the operator runs today; one key record per version of that part, since a second record for the same version would tag the agents given it; admission under its published policy; its share released only as the specification says	P3, P4, P9, P10
T-Rel (relay)	The relay strips network addresses, and shuffles and delays submissions	P11

The properties of §8 use no assumption beyond these; none of them has been checked by an independent audit of this implementation. On BLS12-381, the curve in production, the computational assumptions do not reach the 128-bit level the specification requires; §3.1 states the level and the specified move to BLS24-509.

03

The three-party construction

> One signature over hidden claims, one proof per question, bound to one challenge.

Figure 2 follows one identity credential from issuance to a decision. The issuer identifies the person by its evidence route and issues the credential blind: the holder commits to its secrets, proves that the commitment is well formed, and receives a signature over its claims and over the committed secrets that the issuer never sees. Every 30 seconds the issuer writes a status commitment that covers its accumulators, against which every credential it has issued and not removed can prove membership. When the holder wants a service, the verifier reads the record layer and sends a challenge signed with one of its registered request keys; the holder has brought its status witness forward from the issuer's public status log and answers with one zero-knowledge presentation bound to that challenge; the verifier reads the record layer again, checks the proof and every condition of §3.6, consumes the nonce and records its decision.

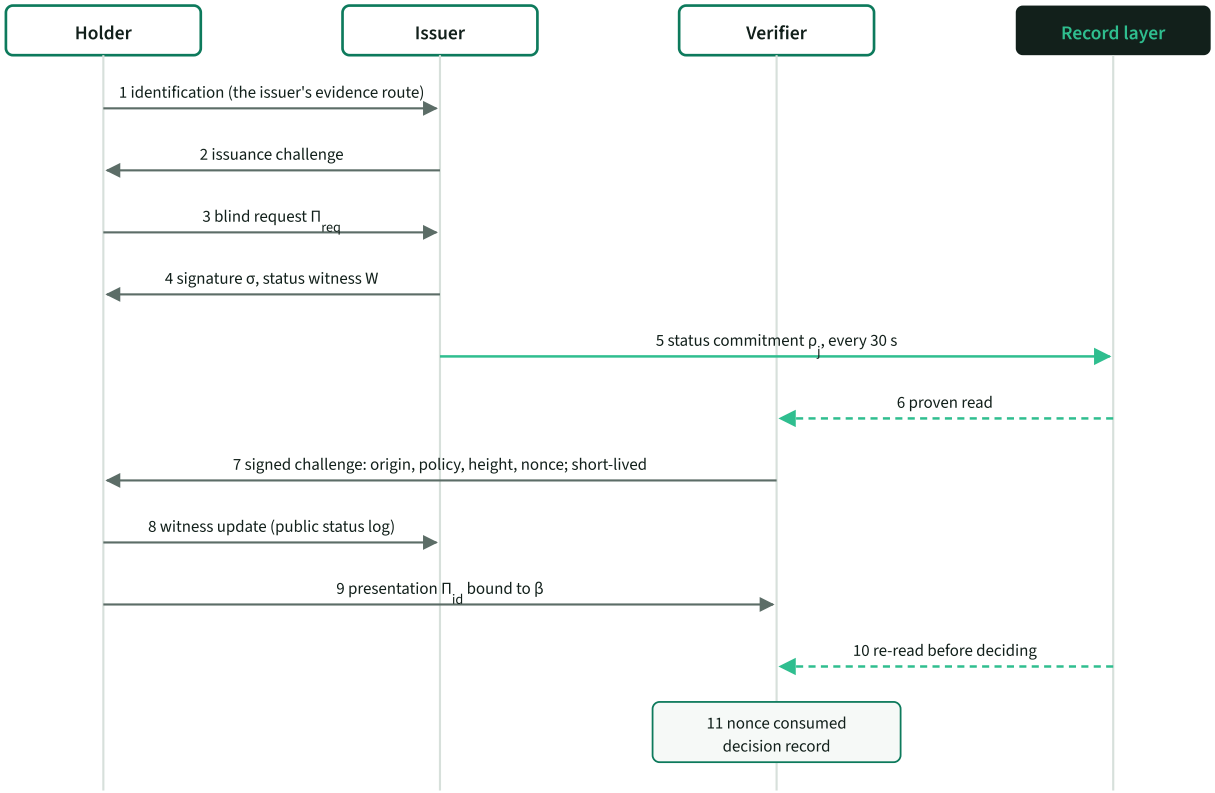

Figure 2 · One credential from issuance to a decision. Nothing in steps 3 to 11 is written to the record layer except the issuer's commitment. Dashed arrows are proven reads.

3.1 Schemes, and why anonymous credentials

Table 4 names every scheme and primitive the construction uses, with its role and its reference. The curve in production is BLS12-381 throughout (the CFRG pairing-friendly curves draft); the specification moves every pairing-based statement to BLS24-509 (below). The libraries are named without versions, and the References give where each is published: AnonCreds v2 (its credx implementation) for the credential, Bulletproofs for the range proofs, Merlin for the transcripts, and gnark-crypto for the record layer's verifier of the self-revocation proof, which uses the project's own implementation of the Merlin transcript, checked against the reference one. BBS was proposed by Boneh, Boyen and Shacham; Au, Susilo and Mu, and Camenisch, Drijvers and Lehmann, proved BBS+, the variant

whose signature carries a further scalar; Tessaro and Zhu proved the one-element, one-scalar variant, the form used here: in the standard model for a uniformly random per-signature scalar and a signer that sees the messages, and in the algebraic group model for blind issuance, and note that the scalar may instead be derived by a pseudorandom function of the whole message; how the scalar is derived here is stated in Table 3.

Table 4 · Schemes and primitives.

SCHEME	ROLE IN THE PROTOCOL	REFERENCE
BBS signatures, in the AnonCreds v2 object model	The issuer's signature over the credential's claims; the holder's proof of knowledge of it over hidden claims, revealing only the issuer	The BBS Signature Scheme (IRTF CFRG draft); Tessaro and Zhu 2023; AnonCreds v2
Blind issuance with a holder link secret	The holder commits to a secret that the issuer signs without seeing it; every proof proves knowledge of it	AnonCreds v2
VB20 dynamic accumulator	Non-revocation (status) and agent scope membership, proven without naming the member	Vitto and Biryukov 2020
Pedersen commitments	A commitment to a hidden claim inside a proof, linked to the signature by a shared response	Pedersen 1991
Bulletproofs range proofs	Threshold predicates: grade, expiry, age, custody rank, transaction limit, equality of the unit	Bünz et al. 2018
Fiat–Shamir with Merlin transcripts	One challenge for the whole conjunction; the binding digest is the transcript's nonce	Fiat and Shamir 1986; Merlin; STROBE
Hash-to-curve and hash-to-field	Generators with no known discrete logarithms; the one-time code	RFC 9380
Pairwise pseudonym	A stable identifier of one agent credential at one verifier, uncorrelated across verifiers	Pseudonym systems (Camenisch and Lysyanskaya 2001); U-Prove (Brands 2000); BBS per Verifier Linkability (IRTF CFRG draft)
The self-revocation proof	A BBS proof with the one-time code revealed, accumulator membership and an ElGamal encryption of the status reference, as coupled Schnorr proofs and two pairing checks, verified in block execution	Schnorr 1991; Tessaro and Zhu 2023; Vitto and Biryukov 2020
ElGamal in G_1 with a proof of correct encryption	The status reference encrypted to the issuer in a self-revocation; the dispute ciphertext under the joint key	ElGamal 1985
Additive joint key with Schnorr proofs of possession	Neither the issuer nor the Foundation alone can open a dispute ciphertext, and neither can choose its part to cancel the other's	Schnorr 1991
Decryption shares with Chaum–Pedersen proofs	Verifiable partial decryption, bound to the claim and the presentation	Chaum and Pedersen 1992
Ed25519	Status commitments, emergency commitments, the verifier's signed challenge, the Foundation's admission grants and its dispute-key record	RFC 8032

SCHEME	ROLE IN THE PROTOCOL	REFERENCE
HMAC-SHA-256	The derivation of the MintID Foundation's part of the dispute key from a non-exportable key; the issuer's cover removals and cascade draws in its public status log	RFC 2104; FIPS 180-4
SHA-256 Merkle tree with the leaf and node prefixes of RFC 6962	The issuer's status commitment over its status entries, accumulators, definition and encryption keys; that a root names one list of leaves is pending audit (the implementation audit)	RFC 6962 (obsoleted by RFC 9162); FIPS 180-4
Binding digest	A collision-resistant digest of the verifier's signed challenge, used as the proof's Fiat–Shamir nonce	SHA-256 (FIPS 180-4)

SHA-256 is the hash of every generator MintID derives, of every digest and of the status-commitment construction. The credential library maps the issuer identifier, the status reference and the scope reference to scalars with SHAKE256 and derives the range proofs' generators with SHAKE256 (FIPS 202); every transcript, the self-revocation transcript that block execution replays included, is a Merlin transcript over Keccak.

The security level and the curve. The specification requires at least 128 bits of classical security: against an adversary that sees every public value, controls any number of holders and verifiers, obtains blind signatures on messages of its choice and runs up to 2^{128} operations, the best known attack on every assumption of the construction must cost at least 2^{128} operations, and a term that rests on a number-field-sieve estimate in the target group at least 2^{132}. BLS12-381, the curve in production, does not meet it: its target group is estimated at about 126 bits, generic discrete logarithms in its groups at about 127, and the q-type assumptions under BBS signatures and the accumulators lose about half a bit each time the signatures or updates under one key double, to about 110 bits at 2^{30} signatures. The specification moves every pairing-based statement to BLS24-509, the curve that the European Digital Identity Wallet's exploratory specification for zero-knowledge proofs over multi-message signatures (TS14) names, with a BBS ciphersuite that MintID defines on it. Its target group is estimated at about 193 bits, an estimate whose parameters an external cryptographer is yet to check, and its signatures and accumulators stay far above 128 bits at any usage a key can reach, so that, with every reduction into its scalar field widened as the specification requires, the level of the system becomes the conventional 128-bit level of SHA-256, the transcripts and Ed25519, the last counted in that category as the specification states. As hygiene, the specification caps at 2^{40} per key, a figure that only an amendment of the specification changes, the signatures that each version of an issuer's credential-signing key gives out: above it the issuer refuses to sign and asks for a rotation of the key, which goes with a clean regeneration of the chain (§6.4). It also has the issuer count the steps that each accumulator key takes against the same figure and refuse none, since a step can be the removal of a revoked credential: at the figure the issuer takes the step, warns its operator and asks for a rotation of that key. On BLS24-509 every pairing term stays at 184 bits or more at that count. Both counts are in production, on BLS12-381: the count of signatures, and the count of steps for every accumulator key, both the keys of the status accumulators and the key of the scope values' accumulators (§4.4). The move has not started; it carries no date, is built before the implementation, self-revocation and dispute-ciphertext audits so that each reviews the final curve once, and reaches a deployment by a clean regeneration of the chain (§6.4). The project found no implementation of BLS24-509 that it judges fit for production and has decided to write the curve code itself, in Rust and in Go, as open-source code; that code would run in every holder and in block execution, and it is an audit item. No implementation of the protocol on the new curve exists to measure. An estimate, from the measured cost of the curve's arithmetic applied to the operations each proof performs, puts proofs and verifications at about 2.6 to 3.4 times today's cost and presentations 35 to 43 per cent larger. On that estimate native holders and verifiers keep a wide margin; a browser holder on a device four times slower than a high-

end laptop would take 7.9 to 10.9 seconds from challenge to decision for an identity presentation, on a ten-second lifetime in the central estimate and over it in the high one, and 11.7 to 17.6 seconds for an agent's presentation, over it, while each fits a thirty-second lifetime with at least 12 seconds to spare, which is why the presentation lifetime is thirty seconds (§3.5); the record layer's check of a self-revocation proof still fits within a block, its per-block limits an open decision. Nothing in the construction is post-quantum: an adversary able to compute discrete logarithms on these curves would forge credentials and proofs, and could open recorded ciphertexts and link an agent's pseudonyms after the fact, including those kept in block history.

Why anonymous credentials. The protocol needs three things that a signed list of claims cannot give. It needs presentations that cannot be linked to one another, across sessions and across verifiers, through their contents. It needs predicates without values: a verifier that asks whether a grade is at least A2 must learn that and not the grade. And it needs non-revocation without a revocation identifier, so that a verifier's check of a credential's status tells no one which credential is being checked; the holder updates its witness from the issuer's public status log, whose fetches name no credential (§3.7). Salted-hash selective disclosure, as in SD-JWT (RFC 9901) and SD-JWT VC, and the mdoc format (ISO/IEC 18013-5) reveal the same issuer signature and digests in every presentation of one credential, so that a verifier, alone, with another verifier or with the issuer, can link presentations of that credential unless the holder uses a fresh credential for each presentation; RFC 9901 discusses these limits, for all salted-hash approaches, in its own section 10.1 on unlinkability. A BBS signature with a proof of knowledge over hidden claims gives all three properties under the assumptions of §2.5, at the cost of a credential format of its own. MintID does not claim interoperability with SD-JWT VC or mdoc credentials. Its transports follow open standards: OpenID for Verifiable Credential Issuance for issuance and OpenID for Verifiable Presentations for presentation, each carrying the MintID credential format.

No primitive here is new. Three compositions are: the self-revocation proof, which re-implements the BBS proof of knowledge outside the credential library and applies the same form to the accumulator witness (§3.8); the pseudonym statement with the identity point as blinding generator (§4.5); and the dispute ciphertext, a verifiable encryption under an additive two-party key with proofs of possession and verifiable decryption shares (§4.6). The composition of the presentation itself, how its statements and bounds are derived and checked, is the question of the implementation audit. All are audit items. For each of the three the project has written its own argument, not yet published and not reviewed outside the project: the soundness of the self-revocation proof and its witness indistinguishability among the credentials of one issuer and profile, with a simulation that uses the issuer's keys (§3.8); the soundness and zero knowledge of the pseudonym statement with the identity point as blinding generator (§4.5); and a draft argument for the dispute ciphertext's composition (§4.6). The audit scope of the dispute ciphertext is written, and that audit is not yet commissioned (§11).

The designs decided or specified beside the curve add no primitive: the derivation of the signature scalar changes how the credential library signs and is an item of the implementation audit (Table 3); the public status log uses the accumulator's update from public values, which the scope sets already use (§3.7); the record of an issuer's fault adds no proof type (§3.8); and the custodian's co-signature of the Foundation's key record is an Ed25519 signature (§4.6).

3.2 The credential

The issuer signs, with one BBS signature, a vector of claims: a status reference, the holder's link secret, the issuer identifier, the assurance grade, the issue and expiry instants, the date of birth, a profile version and the one-time code that later makes self-revocation possible. The link secret and the one-time code are signed blind. Only the issuer identifier is revealed to a verifier; a self-revocation publishes the one-time code, and a mint or a relay discloses the status reference to the issuer alone. In the notation of §2.4 the identity credential is σ on $m_{\mathrm{id}} = (y, s, I, \gamma, \tau_{\mathrm{iss}}, \tau_{\mathrm{exp}},$

τ_{dob}, N), with its profile-version claim, under the issuer's identity key X_{id}. The agent credential (§4.1) is a second profile with a vector of its own under a key of its own. Table 5 lists the claims of both and how a verifier sees each.

Table 5 · The credentials.

CLAIM	IDENTITY PROFILE	AGENT PROFILE	HOW A VERIFIER SEES IT
y, status reference	yes	yes	Inside a proof only (accumulator membership; for an agent, the dispute ciphertext)
s, link secret	yes	yes	Never
I, issuer identifier	yes	yes	Revealed
g, grade	the identification's grade	the backing grade set at mint	Predicate only
τ_{iss}, issue instant	yes	yes	Never
τ_{exp}, expiry instant	yes	yes	Predicate only
τ_{dob}, date of birth	yes	no	Predicate only
N, one-time code	yes	yes	Never, outside a self-revocation
λ, transaction limit	no	yes	Predicate only
u, unit of the limit	no	yes	Predicate only, as equality
κ, key-custody rank	no	yes	Predicate only
z, scope reference	no	yes	Inside a proof only (scope membership)
v, principal's age floor	no	profile v2 (built, not issued)	Predicate only

The issuer is trusted for every value it writes. What a credential says about the identification behind it is stated in §5.1.

3.3 Blind issuance and the link secret

The holder draws the link secret s and computes the one-time code $N = H_p(s)$. It sends the issuer a commitment to both and a proof that the commitment is well formed, bound to the issuer's own challenge, so a request made for another challenge is refused. The issuer signs the full claim vector over the committed values without learning them.

Π_{req}, the issuance request.

Public: the commitment R and the issuer's issuance challenge. Witness: s, N.

1. $R = s \cdot h_S + N \cdot h_N$;
2. the proof is bound to the issuer's challenge.

Not proven: $N = H_p(s)$.

Issuance travels over OpenID for Verifiable Credential Issuance, in production in the shape of its 1.0 Final text; the issuer still answers the pre-1.0 shape for clients that have not moved, until a later release removes it. A credential becomes usable once a status commitment that includes it is final on the record layer; no record-layer transaction says that a credential was issued.

The issuance request carries no separate blinding factor: the link secret is hidden from the issuer because the one-time code committed beside it is a random-oracle output of the secret. Once a self-revocation publishes that code, the secret stays hidden from the issuer under DL in G_1, and linking it to a pseudonym rests on DDH in G_1. That the code is the hash of the secret is computed by the holder and certified by the issuer's signature; no proof establishes it. A holder that departs from that relation weakens the hiding of its own secret.

3.4 The presentation

An identity presentation is one zero-knowledge proof, under one Fiat–Shamir challenge, of knowledge of the issuer's signature over hidden claims with only the issuer identifier revealed, of membership of the hidden status reference in an accumulator of the issuer covered by its current status commitment or by a recent one that §3.6 accepts, and, per requested predicate, of a commitment to the signed claim with a range proof on it, all linked by shared responses. The shared response is what makes the conjunction one statement about one credential: the value committed for a range proof is the value under the signature, and the status reference proven a member of the accumulator is the one signed.

Π_{id}, the identity presentation.

Public: X_{id}, Q_{id}, V_{id}, I, Φ, t_{ref}, β. Witness: σ, the hidden claims of m_{id}, W, the commitment randomness.

1. σ is valid on m_{id} under X_{id}, and the issuer claim is I;
2. $\hat{e}(W, Q_{id} + y \cdot g_2) = \hat{e}(V_{id}, g_2)$;
3. for each predicate of Φ, a Pedersen commitment to the signed claim and a range proof: $\gamma \geq \gamma_0$; $\tau_{exp} > t_{ref}$; $\tau_{dob} \leq \vartheta(t_{ref}, n')$;
4. the transcript carries β as its nonce and the statement, so that the proof holds for this verifier, this policy and this instant only.

The predicates of an identity policy are: the grade is at least γ_0; the holder is over n' years; the credential is active, which is conjunct 2 and is present in every presentation; and the credential is unexpired, which every policy asks, a verifier refusing a policy that does not. Each predicate is evaluated at the challenge's issue instant, so a credential whose expiry falls inside a presentation's lifetime can be accepted until that lifetime ends. The verifier receives the issuer identifier, the accumulator the proof was made against, the reference instant and one true value per predicate, and nothing else.

A presentation answers a policy with one credential; presentations that combine several credentials are not part of the protocol. The soundness of the composition (how the statements are derived, the bounds, the verifier's checks) is the question of the scheduled implementation audit.

3.5 The verifier's challenge and the binding

A registered verifier opens a session with a challenge signed by one of its registered request keys. Every presentation proof is bound, as its Fiat–Shamir nonce, to a digest of the verifier's origin, its identifier, its request key, a fresh single-use nonce, the digest of the whole signed request (policy, accepted issuers, expiry) and the finalised height the verifier read. A presentation is valid only for its lifetime, counted from the challenge's issue: the presentation window, one constant of the presentation format, the same for every presentation that answers a verifier challenge and for the issuer's mint challenge, which no request, configuration or binding sets; an answer at or after the expiry is refused. The lifetime is thirty seconds. The lifetime bounds how long a challenge may be answered; replay is refused by the single-use nonce and the binding, and status freshness is the verifier's check at the decision (§3.6), so the lifetime lengthens no revocation bound (§6.3).

The binding makes a proof a statement about one decision. A presentation made for one verifier, one profile or one policy never verifies for another, because each of them enters the digest that the proof's challenge depends on. The verdict is a statement about one decision, and replay is refused by the nonce, never by the bytes. A presentation has exactly one encoding, which every verifier requires before it reads any value, and the presentation digest is computed by verification over the bytes it accepted, so a claim names an agent presentation by its digest; its replay too is refused by the nonce.

The holder library answers only a challenge signed by a key from a source its host supplies; the store of trusted verifiers, and every consent prompt, are the host's. The library is headless by design: it decides whether a challenge is well signed by a key it was given, and leaves to the application that embeds it how the person chooses which verifiers to trust. A host that reads the verifier's record from a node, rather than from a store it keeps, shows that node which verifier it is about to answer. Whether the party asking is the origin the challenge names is also the host's check: the library verifies the signature, not the channel, so a party that relays a verifier's genuine challenge to a holder in real time and returns the answer to that verifier is stopped only by the host (§8.8). On the x402 transport with Web Bot Auth, an answer submitted under a signing key other than the one its challenge was issued to is refused, and relaying the agent's whole signed request unchanged remains a matter of the channel (§4.3; released, in the x402 middleware and the agent clients; built and switched off in the verifier service in production).

The lifetime is measured on the verifier's clock; the holder library does not check the reference instant against its own, and has no rule against a verifier that varies thresholds or reference instants across sessions (§8.4).

3.6 What a verifier checks

A verifier accepts a presentation only when every condition below holds, in this order. A failure at any step refuses the presentation, and the nonce is consumed only when every earlier step has passed.

1. The verifier, its origin and its request key are active on the record layer.
2. The challenge is the verifier's own, unused, and answered within the presentation lifetime (§3.5).
3. The proof verifies under the binding digest.
4. Every requested predicate is proven.
5. Just before deciding, the verifier re-reads the issuer's state and status commitment and accepts only values proven against the state commitment of a node it designates. The issuer must be active and its commitment current, or a recent one that no emergency commitment followed (§3.7).

6. For an agent, the pseudonym and the dispute ciphertext were checked with the proof (step 3), and the verifier has written its dispute retention record; it refuses the presentation when it cannot.
7. Last, the nonce is consumed once, atomically, in the verifier's own durable store.

The acceptance logic exists in one implementation, offered embedded or as a service; client libraries carry the request and the answer and decide nothing, so the acceptance conditions are the same for every integrator; one that uses a deployment operated by another party relies on that party to run them.

Replay protection is per verifier; a presentation made for another verifier is refused by its binding. The binding to the status commitment is indirect: the proof binds the accumulator and the issuer definition it used, and the verifier then requires them under the current commitment or a recent one that step 5 accepts. Verification is online and fails closed: the verifier needs live record-layer state for every decision and refuses when it cannot read and prove it, so a silent issuer takes its credentials offline within the maximum accepted age of its commitment (§3.7).

3.7 Status commitments and non-revocation

The issuer commits to the status of its credentials in one signed commitment at every heartbeat of 30 seconds; the commitment covers the issuer's accumulators, its definition and its keys (Table 2, ρ_j), and nothing per holder reaches the record layer. A holder proves its credential unrevoked by proving that its hidden status reference is a member of an accumulator covered by the current commitment or by a recent one that §3.6 accepts. The accumulator records removals, not members: the issuer can make a witness for any reference it has not removed, so a commitment says nothing of how many credentials an issuer has issued.

Status commitment and emergency commitment (signatures, not proofs).

1. $\mathrm{Sig}_{rk_l}(l, \rho_j, j, \rho_{j-1}$, generation instant, validity window, $\alpha_j)$, accepted by the record layer only under an active key of an operational issuer, with j strictly increasing, α_j neither below the previous count nor above the codes recorded, every code recorded at least T_G before the block time acknowledged, and the time bounds of §6.3 held;
2. Sig_{ek_l}, likewise, for an emergency commitment, which carries the issuer's last acknowledgement forward, is held to the same rule and takes effect at its finalisation.

The accumulator moves only when it must. Issuing a credential never invalidates another holder's witness; only revocations, and the cover moves of the agent accumulator at every heartbeat, change an accumulator. A verifier refuses, fail closed, a commitment past its validity, one superseded beyond its height policy or followed by an emergency commitment, an inactive issuer, and any decision for which the state cannot be read and proven. An emergency commitment is signed with the issuer's separately scoped emergency key, alone, and takes effect at its finalisation. The record layer limits emergency commitments per block; no other check bounds the use of the emergency key, and the compliance council that would review it is built into the chain and not seated. The maximum age of a commitment a verifier accepts is 45 s at the reference values and 180 s on the single-node production chain (Appendix B).

Which recent commitments a verifier accepts depends on the profile, because an agent accumulator moves at every commitment and an identity accumulator only when an identity credential leaves. An agent presentation is checked against the issuer's accumulator and definition under the commitment it names, which must be the issuer's latest finalised commitment or the one that commitment superseded. An identity presentation is checked against the identity accumulator under the issuer's latest finalised commitment, and the decision rests on that commitment, whatever commitment the presentation names; a presentation that does not verify against that accumulator, which has moved since the holder's witness, is refused as stale. A session or an access token built on an accepted

presentation follows the rule of its profile (§4.3): for an agent, the commitment its acceptance rested on, while it is the latest finalised commitment or the one the latest superseded; for identity, the latest. Every commitment these rules accept is one that the rule above accepts, so no revocation bound depends on the profile (§6.3). At the reference values a superseded commitment stays inside its validity window for 15 seconds, and with the holder's fetch schedule below an estimate by simulation refuses as stale about one attempt in three of an agent whose proof takes 11 to 14 seconds, and about one in two of one whose proof takes 17 to 21 seconds; a deployment with a longer maximum accepted age refuses fewer.

The issuer does not see the proof, and nothing a verifier sends at a decision names the credential. A verifier that reads the issuer's published definition at each decision, as the reference verifier service does, shows the issuer when it decides, though not which credential. Status witnesses, like scope witnesses, are brought forward from a public log: after each commitment the issuer publishes, for each profile, the removals of that commitment from its accumulator, in batches of a fixed size completed with cover removals, every commitment carrying at least one batch for each agent profile, the identity profile having a batch only in a commitment that removes an identity credential, and a commitment whose due removals exceed one batch carrying several, all committed under the status commitment by an entry that chains every earlier publication; a holder updates its own witness from that publication, which it checks against a commitment it reads with a proof, and from nothing that names its credential. A holder fetches that publication on a schedule independent of presenting and common to the holders of a profile, an agent holder at each commitment's expected generation and an identity holder at the multiples of a fixed period, each after an offset common to every holder and with a delay drawn afresh, and presents with what its last fetch left; as released, a holder also fetches when its software starts, so that a holder started in order to present shows when it started. The removed accumulator elements, and which commitments remove an identity credential, are public, and the server of the publication, and the node that serves the holder's status reads, learn which network addresses fetch or read and when; on production the issuer's operator runs both. That the padding hides from an observer of the commitments how many removals were due is pending audit (the privacy review of the public status log). A removal in the publication is the issuer's assertion: the issuer, which holds the accumulator secret, can still remove one credential and can still make a witness for any element, so removal is not provable in this accumulator. A holder whose issuer publishes no such log, whose last update is older than the publication the issuer keeps, or whose host enables it, asks the issuer for its witness, per credential: the issuer then learns when that credential is active and, joined with the time of a verifier's definition reads, can learn where it presents, and it can withhold the witness. Freshness is the verifier's check of the commitment, never something the proof alone establishes.

3.8 Holder self-revocation

A holder self-revokes (Figure 3) by publishing the one-time code signed into its credential, with a proof, made without a circuit, a SNARK or a trusted setup, that the code belongs to a live credential of the named issuer, and with the credential's status reference encrypted to that issuer. The record layer verifies the proof in block execution and records the code; the issuer removes the credential at its next commitment, a removal the record layer cannot see, and the record layer refuses every commitment of an issuer that has not acknowledged a recorded code within the grace, so that all its credentials stop verifying. Holder self-revocation is in production. It is itself a governed value of the record layer: a recorded decision can switch it off, and on the single-node production chain that decision is a governance proposal voted by the single validator, which the operator runs.

Π_{srp}, holder self-revocation.

Public: X_π, Q_π, V_π (covered by a recent ρ_j), K_{sr}, I, N, β_{sr}, (E_1, E_2). Witness: σ, the hidden claims, W, r.

1. σ is valid on the profile's claims with I and N revealed (the BBS proof in the form of Tessaro and Zhu, re-implemented);
2. y is a member of V_π (the same form, applied to the witness);
3. $E_1 = r \cdot g_1 \wedge E_2 = y \cdot h_m + r \cdot K_{sr}$;

one y across 1 to 3, by a shared response. Verified in block execution by two pairing checks and four Schnorr equations under one challenge. Not proven: $N = H_p(s)$.

The proof is bound to the record layer, the issuer, the profile, the code and one recent commitment of the issuer, so it replays neither across chains nor across regenerations. The holder needs no account: a privacy relay carries and pays for the submission, strips the network address before its application sees it, shuffles a batch and delays it over more than one block. A holder can self-revoke without contacting the issuer only against a commitment still in the issuer's ring of recent commitments whose accumulator matches the witness it cached at its last refresh. An identity credential's cached witness lasts until the issuer's next removal of any credential, plus the ring; an agent's lasts only the ring, about four minutes on the single-node production chain, because the agent accumulator moves at every heartbeat. After that the holder needs a fresh witness from the issuer, which sees the request and can withhold it. The one-time code links to no credential identifier, no holder, no sibling, no pseudonym and no past presentation; two self-revocation proofs are unlinkable except through the code, and only the issuer, which decrypts the status reference, learns which credential a code ends.

The design puts the proof on the record layer, rather than at the issuer, for one reason: so that the effect of a self-revocation does not depend on the issuer's cooperation. Because the record layer checks the proof, a recorded code is, under the soundness of a proof whose audit is scheduled, one that only a holder of the signature and of a current witness of a credential carrying that code could have produced, a party that copied the link secret included (§3.9); its ciphertext names that credential to the issuer, and the acknowledgement rule can bind the issuer to it.

The limits of holder self-revocation are these. The relation between the one-time code and the link secret is certified, not proven (§3.3). Nothing at issuance makes a code unique to one credential, so a recorded code does not by itself show which credential the record revokes. The holder library, as released, reports a self-revocation as done only when the record carries the ciphertext its own message carried and reports any other record of its code as not confirmed, on which the holder asks its issuer to revoke the credential (§6.3); proving the code's relation to the link secret inside the proof would prevent a record of the code under a credential whose link secret does not give that code, and is deferred until an external review of the proof. One of the proof's three statements, accumulator membership in the form of the signature proof, is a new composition, and the signature proof is re-implemented outside the credential library (§3.1); the project's soundness and witness-indistinguishability arguments for the proof are written, not yet published and not reviewed. The record layer enforces that the issuer acknowledges a code, not that it removes the credential: an issuer that acknowledges without removing keeps the credential verifying, invisibly to the record layer, and only the holder can detect it. A self-revocation record is kept for an evidence window of 24 hours after the issuer's fold; within it, a holder whose credential the issuer left out of its committed removals (§3.7) can prove on the record layer that the credential the issuer folded is still a member, and the record layer then records the issuer's fault, freezes the issuer's live bond if it is active, and signals the protocol's authority; the record rests on the soundness of the self-revocation proof, whose audit is scheduled. The issuer corrects the fault within a governed period by removing the credential and recording the correction; an uncorrected fault is a ground for suspension, which the protocol's authority decides and the record layer never does. Until the compliance council is seated, the

admission policy suspends an issuer whose fault stays uncorrected, a criterion that leaves no discretion: the MintID Foundation decides that suspension on that ground, and whether a bond frozen on the fault is unfrozen or slashed, and the operator executes each decision; the correction period is 24 hours at the reference values. On the single-node production chain the only issuer is the operator's own (§9.3); whether a related party may execute its suspension is a counsel item, and open. An issuer that lists the element among its removals and keeps it a member leaves no such evidence. If the issuer withholds fresh witnesses, a holder whose cached witness has left the ring cannot prove; for an agent, that is any holder that has not refreshed within the last eight commitments. A holder updates its witness from the issuer's public status log, and only an issuer that publishes no log, or a holder whose last update is older than the publication the issuer keeps, leaves the holder dependent on the issuer's witness. The single validator can delay or censor a submission, and can switch self-revocation off. The relay strips addresses at its application, and whatever stands in front of it is not covered. A holder may reach the relay through a transport of its own that hides its address, such as Tor; a transport provided with the holder library that hides the holder from the operator (Oblivious HTTP, RFC 9458, through a party other than the operator, or a Tor onion service), and relays run by others, are due before the first node the operator does not run serves holders, a stage that carries no date. On production one operator runs the relay and the issuer, behind one load balancer that keeps no access log by that operator's configuration, so their separation rests on that operator (T-O). The holder's confirmation read of its own self-revocation is not proven. A reproducibility run of the record layer's verifier on its native production platform is pending.

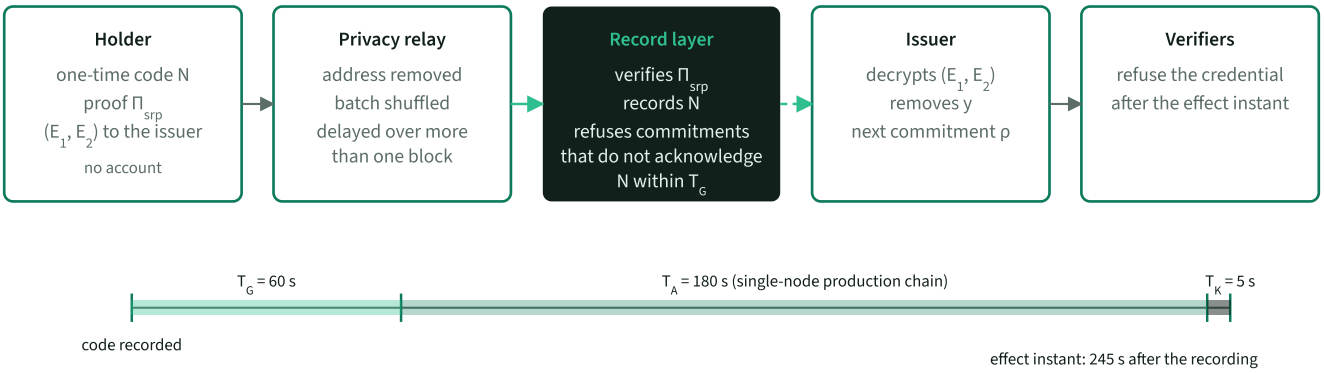

Figure 3 · The one proof the record layer verifies. The holder needs no account; the issuer cannot leave the code unacknowledged without its credentials going stale, and its removal of the credential rests on the issuer. The dashed arrow is the issuer's read of the recorded code.

3.9 Renewal and recovery

A credential lasts no longer than the identification behind it; its expiry is the identification's reuse window for its grade. Renewal is a new identification and a new credential with fresh secrets, not publicly linkable to the one it replaces; the issuer links them in its records. A lost or compromised link secret is recovered only by a new identification and a new credential; there is no social recovery. Whoever copies a link secret can present with that credential until it is revoked; self-revocation is the holder's answer.

04

Agents in the three-party construction

> An agent is a holder of a credential its issuer minted from a person's credential; no party and no record is added.

The agent credential uses the same scheme, the same issuer and the same verifier; it adds a profile, a mint, three predicates, a pseudonym and a ciphertext, and nothing to the record layer per agent. The choice to derive agents at the issuer, rather than to register them anywhere, follows from the invariants of §2.3: a public registry of agents would record who delegated to whom, and the record layer would grow with people; the protocol keeps that link only in the issuer's accountability escrow.

4.1 The agent credential

The agent credential is σ on $m_{ag} = (y, s, I, \gamma, \tau_{iss}, \tau_{exp}, N, \lambda, u, \kappa, z)$, with its profile-version claim, under the issuer's agent key X_{ag} (Table 5). It carries no date of birth: in its place it carries the transaction limit λ and its unit u, the key-custody rank κ and the scope reference z. The grade is the backing grade the issuer set at mint, bounded by its policy and by the principal's own grade. The expiry is the earliest of the lease, the principal's expiry and the end of the scope's validity.

Human-backed is not a claim: a valid signature under the agent profile is what the issuer gives only to an agent it minted from a principal credential. Agent profile v2 adds the principal's age floor v, so that an agent can prove that the person behind it is over a given age without any date reaching the agent or the verifier; it is built, not issued. Everything the agent credential says about its person is attested by the issuer at mint, under the limits of §4.2.

4.2 Minting from a principal presentation

The issuer mints an agent credential from a presentation of the principal's identity credential that proves a grade at least the agent's and that the credential has not expired, and that discloses the principal credential's status reference to that issuer only; the issuer checks the principal's standing against its own registry, records the link from the agent to the principal credential in its accountability escrow, and issues the agent credential blind, with a fresh link secret, a fresh status reference and a fresh scope reference. The mint challenge comes from the credential's own issuer, and the holder refuses any other. Minting writes nothing to the record layer beyond an aggregate count of mints per issuer (Figure 4). In the design an agent carries a mint fee and a lease fee, which the companion describes; on the single-node production chain their amounts are placeholders in a valueless denomination that the operator holds.

Π_{mint}, the mint presentation.

Public: X_{id}, I, y_p (the principal credential's status reference, disclosed to the issuer only), γ_a, t_{ref}, the issuer's mint challenge. Witness: σ_p, the other hidden claims of the principal's m_{id}.

1. σ_p is valid on m_{id} with issuer I and status reference y_p;
2. $\gamma \geq \gamma_a$;
3. $\tau_{exp} > t_{ref}$;
4. optionally, $\tau_{dob} \leq \vartheta(t_{ref}, n')$, for an agent profile v2 floor.

No accumulator conjunct: the issuer checks y_p against its own registry, and y_p is the value σ_p signs: a verifier refuses a presentation whose disclosed claims are not the values its signature proof proves (§11). No conjunct links Π_{mint} to the agent's Π_{req}, which is made with a fresh s'.

No zero-knowledge statement links the principal's presentation to the agent's issuance, and no agent presentation makes a statement about the principal's credential. That an agent was derived from a principal credential, that its grade does not exceed the principal's, its age floor, the cap on a person's active agents and the freshness of the identification are enforced by the issuer at mint; that the principal's credential is still unrevoked when the agent presents rests on the issuer's cascade (§4.8). A verifier relies on the issuer for all of them: what it verifies is the issuer's signature under the agent profile, which is the backing the specification requires. The issuer counts the cap over the identification sessions it links to one person; on the wallet route of §5.4 it receives no signal that two unlinked sessions are one person's. The mint is not anonymous to the issuer, by design: the issuer reads the principal's record by the disclosed reference, which Π_{mint} binds to the credential it proves, so the grade and expiry conjuncts of Π_{mint} hide nothing from it, and the link it records is the one the dispute cascade later needs (§7.3). The issuer ties the agent's blind request to the principal's mint presentation through its own issuance session only; whoever obtains that session can obtain the agent, so the session's transport is part of what the mint rests on.

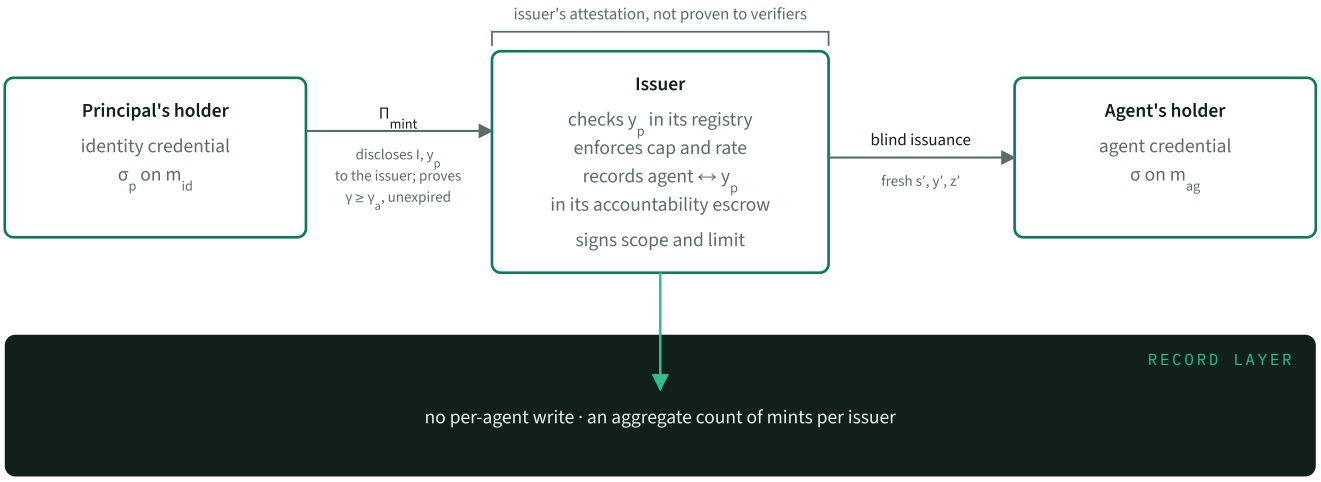

Figure 4 · The issuer derives the agent; what it checks at mint is attested by its signature, not proven to the verifier.

4.3 The agent presentation

An agent presentation is an identity presentation over the agent profile with three further predicates and two mandatory components. The predicates are: key-custody rank at least κ_0; transaction limit at least λ_0 in the unit the

policy names, the unit proven by equality and not revealed; and membership of the scope reference in the set of each named scope value. The mandatory components are the pairwise pseudonym for the verifier (§4.5) and the dispute ciphertext (§4.6). Every agent policy also asks that the credential be unexpired. The verifier receives what it receives for an identity presentation and, in addition, the scope values its policy named, the pseudonym, and a ciphertext it cannot read.

Π_{ag}, the agent presentation.

Public: X_{ag}, Q_{ag}, V_{ag}, Q_{sc}, $\{V_\omega\}$ for the values Φ names, I, Φ, t_{ref}, β, v, P_v, K_d, (C_1, C_2). Witness: σ, the hidden claims of m_{ag}, W, $\{W_\omega\}$, ζ, the commitment randomness.

1. σ is valid on m_{ag} under X_{ag}, with issuer claim I; this is what "human-backed" asserts, a valid signature under the agent profile key, with no statement of its own;
2. y is a member of V_{ag};
3. $\tau_{exp} > t_{ref}$, in every agent presentation; as requested: $\gamma \geq \gamma_0$; $\kappa \geq \kappa_0$; $\lambda \geq \lambda_0$ and $u = u_0$ (equality, as a range with equal bounds); for profile v2, $v \geq n'$;
4. for each scope value ω named, z is a member of V_ω under Q_{sc};
5. $P_v = s \cdot h_v$;
6. $C_1 = \zeta \cdot g_1 \wedge C_2 = y \cdot h_d + \zeta \cdot K_d$;
7. the transcript carries β, the statement, h_v, K_d and (C_1, C_2) before the challenge.

An agent presentation travels in the transport its verifier uses: OpenID for Verifiable Presentations, an agent-to-agent handshake, a transport profile for x402 payments and Web Bot Auth, or an OAuth 2.0 token exchange (RFC 8693) that turns an accepted presentation into a short-lived access token; each carries the same presentation. On the transport profile for x402 payments and Web Bot Auth, the verifying party issues its challenge to the key that signed the agent's request for it, commits the signed request to that key without carrying a handle on it, and accepts the presentation only in a request signed by the same key; by default it serves no agent that does not sign; the credential carries nothing of the key (released, in the x402 middleware and the agent clients; built and switched off in the verifier service in production). A session or an access token that a transport builds on an accepted presentation is part of that acceptance. An access token that its resource server validates without introspection lasts until its expiry, at most the maximum accepted age of a status commitment after it was issued. Under the specification a session ends when a fresh presentation of its profile would be refused on the status commitment its acceptance rested on (§3.7). In the released x402 middleware that a verifying party runs, on the verifier service's status-commitment check, which is in production, the x402 session re-checks on every reuse the issuer's status commitment its acceptance rested on, and ends as soon as that commitment is no longer the issuer's current one; for the agent-to-agent handshake's session a re-check on every reuse is specified and released, in the verifier library, which no deployment runs (§9.2, item 24). The agent-to-agent handshake is provisional. A verifier that takes a payment may bind the presentation to that one payment by a digest of the payment's terms, drawn fresh for each challenge, written into its challenge and kept only by that verifier (specified; released, in the x402 middleware and the agent clients; built and switched off in the verifier service in production). The payment itself stays on its own rail and is paid from the person's or the agent's own wallet, card or account. The digest carries no payer identifier, and no value of the protocol enters the payment. The binding's profile, accepted and in force, names as carriers the open payment protocols x402 (version 2, which the released middleware speaks when its operator selects it; version 1 is its default), the Machine Payments Protocol, the Agentic Commerce Protocol, the Universal Commerce Protocol, and the Agent Payments Protocol with a Universal

Commerce Protocol checkout; the MintID Foundation may admit another protocol that passes the profile's tests. Naming a carrier states that the digest can be computed for it; no integration with any payment company exists.

Key-custody rank at least κ_0 proves the rank the issuer recorded at mint after appraising the agent's key-attestation evidence. The presentation is not signed by that key: the credential is held by its link secret, a software secret in the holder's custody, and whoever copies it can present at that rank. The limit is a bound signed into the credential, not a count of what the agent has spent.

4.4 Scope

An agent's scope is signed into its credential at mint and proven by membership of a hidden scope reference in the set of each named value, on four axes: action, counterparty, category and place; the transaction limit carries the value bound and the credential's expiry carries the time bound. An agent cannot widen its own scope, and a membership witness borrowed from another agent is refused. The scope reference is an issuer-drawn value, not a list: for every scope value it delegates, the issuer gives the agent a membership witness for that value's accumulator, which granting does not change, so a presentation proves membership of each named value without disclosing the others. A value's accumulator moves only when the issuer withdraws a delegation.

The issuer's scope directory discloses which values of the vocabulary have delegated agents at that issuer. There is no time-window axis and no value window across counterparties in the credential, and the limit bounds each transaction and counts nothing.

4.5 The pairwise pseudonym

$P_V = s \cdot h_V$, with h_V derived by hash-to-curve from the verifier's identifier, proven equal to the signed link secret inside the same proof. Every agent policy carries it, so an agent's presentations to one verifier are linkable by design: a verifier needs to hold one agent to its own record (its history, its limits, a refusal) without learning who stands behind it. The issuer cannot compute an agent's pseudonym at any verifier. The holder library refuses a pseudonym request for another verifier.

Pairwise unlinkability rests on DDH in G_1 with hash-to-curve as a random oracle, and is an audit item. The use of the identity point as the statement's blinding generator is a parameter the credential library's authors did not document; the presentation format fixes the statement's encoding with the rest of the presentation's, and every verifier refuses another (§3.5). The project has written a security argument for the statement with that parameter (soundness, zero knowledge and the transcript); it is written and not reviewed, and the statement remains an audit item. A re-mint draws a fresh link secret and so a fresh pseudonym.

4.6 The dispute ciphertext and the dispute key

Every agent presentation to a registered verifier carries a fresh ElGamal encryption of the agent's status reference under a key that is the sum of an issuer part and a MintID Foundation part, each published with a proof of possession, and a proof inside the presentation that it encrypts the presented credential's reference. The verifier cannot read it, refuses an agent presentation without it, and keeps it, with the presentation, its binding and the pseudonym, in its dispute retention record for the claim window, longer only while a claim on it is open. It is bound to its presentation and cannot be lifted into another presentation or claim. Two presentations of one agent carry unrelated ciphertexts. Opening needs a decryption share of each part, each with a proof bound to the claim and the presentation; neither the issuer nor the Foundation can open a ciphertext alone. It encrypts the agent's status reference, not the person's identity: opening it yields the agent, and the issuer's accountability escrow leads from the agent to the person.

Π_{pop}, possession of a key part.

Public: K_X for $X \in \{I, F\}$, its role and its context. Witness: k_X.

1. $K_X = k_X \cdot g_1 \wedge K_X \neq 0$, bound to the role and the context (a Schnorr proof).

Purpose: no party can publish a part whose secret it does not know, so neither can choose its part to cancel the other's.

Π_{sh}, a decryption share (release specified).

Public: K_X, (C_1, C_2), D_X, the claim, the presentation digest. Witness: k_X.

1. $\log_{g_1} K_X = \log_{C_1} D_X$ (a Chaum–Pedersen proof), bound to the claim and the presentation.

Opening: $M = C_2 - D_I - D_F = y \cdot h_d$, matched by the issuer against its table of the references it issued. With one share only, what remains is an ElGamal ciphertext under the other part.

The dispute ciphertext, the dispute retention record and the record of a claim on the record layer are in production; the release of the Foundation's share after a recorded claim is specified, in the tree of the release and run by no deployment, and the issuer's completion is specified, in production's tree and called by no route (Figure 5). The ciphertext is mandatory because a claim must be able to reach an agent that will not cooperate: a claimant that holds an accepted presentation holds, in it, the only route from that presentation to the agent, and the route needs both key parts.

Together, the issuer and the Foundation could open any ciphertext they hold; the separation of their custody is the guarantee against that, and today it is procedural. The key has two parts, each reserved to its own role, the issuer's and the Foundation's, and no other role may use the Foundation's part; today both parts are kept in infrastructure that the operator runs, and one natural person, the founder named in §9.3, holds the credentials of both roles, so the separation is one of roles within related parties and not one between independent parties. An opening needs both parts and, under the specification, a recorded claim.

From a later clean regeneration, which follows the custodian's key ceremony, the Foundation's part is generated in the custody of a custodian that operates no issuer, chosen by the MintID Foundation, under an authorisation that needs at least two approvers, and the custodian co-signs its key record, so that holders and verifiers refuse a record without that co-signature, and a holder then keeps the first record body it accepts for each version of the part, for the life of its process, and refuses another; the earlier part is deleted at that regeneration, and presentations carry a further presentation version from its first block. The custodian and its contract are open; the two approving parties of its key ceremony are the MintID Foundation and its founder, the operator's beneficial owner, so that, until the compliance council is seated with several persons, one natural person can still reach both parts. The custodian does not separate the openings at the verifiers the operator runs, which today are every verifier on production: there the operator could still obtain an opening through a claim it files, and each such opening is public and counted. The Foundation that chooses the custodian is a related party of the operator (§9.3), and no claim of independence is made.

The ciphertexts sit at verifiers, in their retention records, for the claim window, a governed parameter of the record layer, 90 days at the genesis of the single-node production chain, which the verifier reads with its proof; where the record layer does not carry the window, the verifier's own setting is held within the range the record layer allows the governed one; the length of the window and the data-protection framework of that record are open. The dispute ciphertext is new cryptography; its audit scope is written and the audit is not yet commissioned. The project has

written a draft security argument for its composition, not yet published: confidentiality under the joint key given both proofs of possession, and nothing learned from one decryption share alone, under DDH in G_1 and the random-oracle model. One step of it is open: a party that receives decryption shares for verified presentations holds a decryption oracle limited to them, which IND-CPA alone does not cover, and that it learns nothing from it about other ciphertexts is left to the audit, following the pattern of signed ElGamal encryption. The draft has not been reviewed outside the project. Identity presentations carry no dispute ciphertext.

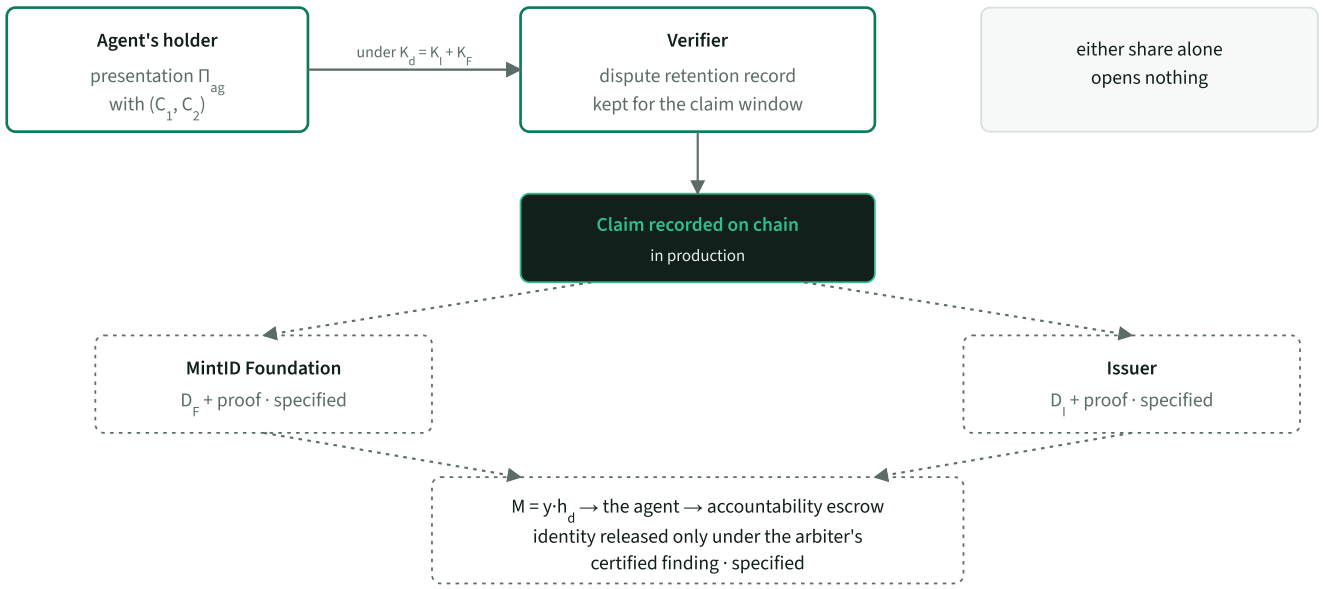

Figure 5 · Solid: in production. Dotted: specified. The ciphertext names the agent; only the accountability escrow leads to the person. The two parts are cryptographically separate; today both are kept in infrastructure the operator runs, and one person holds the credentials of both roles; under the specification, from a later clean regeneration, which follows the custodian's key ceremony, a custodian that operates no issuer, chosen by the Foundation, a related party of the operator, holds the Foundation's part (§4.6).

4.7 The relay presentation

For consented commercial disclosure (§7.4), an agent can relay a relying party's message to its issuer with one proof that discloses its status reference to the issuer and carries its pseudonym for that relying party, so the issuer learns which agent the pseudonym names only when the agent relays; the agent relays and never consents. The statement set is in production; consented commercial disclosure, which it serves, is built and switched off.

Π_{rel}, the relay presentation.

Public: X_{ag}, I, y (disclosed to the issuer only), v, P_v, t_{ref} (the relay challenge's issue instant) and the digest of the issuer's relay challenge. Witness: σ, the other hidden claims of m_{ag}.

1. σ is valid on m_{ag} with issuer I and status reference y;
2. $\tau_{exp} > t_{ref}$;
3. $P_v = s \cdot h_v$;
4. the transcript carries the digest of every field of the relay challenge as its nonce.

No accumulator conjunct: the issuer checks its own registry, and y is the value σ signs, as for Π_{mint} (§4.2).

4.8 Revoking an agent

Three levers end an agent's authority. The first is the agent's own self-revocation through §3.8, the per-agent kill switch, which revokes the credential's authority and does not halt the agent runtime. The second is issuer status revocation, which includes issuer-side revocation by the person on the principal control surface: the issuer's own passkey-authenticated page, which holds no credential, no credential secret and no presenting key; the holder library stays the sovereign path, and the surface is built and switched off in every deployment. The third is the principal-wide cascade: when the principal's identity credential is revoked, the issuer revokes every live agent derived from it, each at its own removal slot, which the issuer draws among the next eight commitments, and none after the last of them. The bound of each path is in §6.3; a session or an access token that a transport built on a presentation accepted before that bound is part of that acceptance (§4.3).

The cascade is issuer policy, not cryptography, and it is the only path by which a revocation of the principal's credential reaches the agents: no agent presentation checks that credential again (§4.2). The record layer enforces that the issuer acknowledges a principal's self-revocation (§3.8); nothing enforces the removal of the agents, and an issuer that omitted the cascade would leave them presenting until their own expiry, which never exceeds the principal credential's. That the removals of a principal's agents are indistinguishable from unrelated revocations is tested by an adversarial campaign and is an audit item. A proof in the agent presentation that the principal's credential is unrevoked is a direction parked until the design of agent profile v3 (§10.2).

4.9 What agents add to the record layer

The agent layer adds no proof type to block execution: the only proof the record layer verifies is the self-revocation proof. From the record layer's view an agent credential is another credential under its issuer's status commitment; the only per-issuer additions are aggregate fee counts.

Agent profile v2 needs one entry in the record layer's table of self-revocation profiles, which is consensus code; the single-node production chain carries that entry, and profile v2 is built, not issued: the issuer has not switched its mint to it.

05

The identification as a parameter

> The protocol fixes what a credential says about an identification, not how the identification is done.

5.1 What a credential says about an identification

A credential says three things about the identification behind it: which accepted issuer performed it (revealed), its assurance grade (never revealed, proven as a threshold) and the end of its validity (never revealed, proven as unexpired, which every policy asks). It carries no route, no provider, no freshness band and no evidence, and nothing on the record layer names or depends on a route.

The protocol depends on no particular evidence route: a document and liveness check, a national eID or another kind of evidence are equally invisible to the record layer, to a credential and to a verifier. What the construction requires of an identification is what the issuer requires before it issues (§5.2), and what it transmits is the grade the issuer grounded. This is why the identification is a parameter of the issuer and not of the protocol: an issuer can add an evidence route without any change to the record layer, the proof core or the verifiers, and a verifier keeps the same policy.

On the classical seller rail a verifier reads an issuer-signed attestation that carries the grade and the freshness band of the identification; that rail is never zero-knowledge.

A verifier that accepts one issuer only, which runs one route only, learns the route by inference.

5.2 What an issuer requires before it issues

The issuer obtains identification through a provider-neutral adapter: each evidence source is a driver behind one interface, and adding one changes nothing in the record layer, the proof core or the verifiers. Before it issues an identity credential, the issuer requires an approved session; a grade grounded by its published policy and inside the grades it may assert; a date of birth read from the evidence source; and a reuse window not yet past. The grade is the issuer's decision over enumerated evidence; an evidence source reports checks and never a grade. Keeping the grade at the issuer, rather than at the evidence source, keeps one grade scale for every route: each route grounds a grade under the issuer's published policy and adds none.

Persons only: an identification that yields no person and no date of birth cannot ground a credential today. The grades an issuer may assert are set at its admission and published on the record layer; the issuer bounds its own grading by them, and nothing else enforces that bound today.

5.3 Assurance grades

A grade measures the strength of an identification, not its age. Table 6 gives the four grades with the validity the issuer applies to each and, beside them, the external frameworks mapped for comparison. A MintID grade is MintID's own; the eIDAS levels are mapped beside it for comparison; a grade is not an electronic identification means under a notified scheme (eIDAS Art. 8) and does not open the AMLR Art. 22(6)(b) route. Entries marked *(counsel)* await qualified legal review.

Table 6 · Grades.

GRADE	VALIDITY	MEANING	EIDAS	ISO/IEC 29115	NIST SP 800-63	AML MAPPING *(COUNSEL)*
A1	12 months	Basic identity binding and issuer verification	low	LoA2	IAL1	customer due diligence
A2	12 months	Stronger document and liveness verification	substantial	LoA3	IAL2	due diligence with biometric liveness, mapped to the baseline proofing of ETSI TS 119 461
A3	6 months	Enhanced due diligence and periodic evidence review	substantial + enhanced due diligence *(counsel)*	LoA3⁺	IAL2 + EDD *(counsel)*	customer due diligence plus screening: A2 plus sanctions and PEP screening at onboarding and at every renewal, and a second independent evidence
A4	3 months	Highest assurance, stricter issuer controls and audit requirements	high	LoA4	IAL3	high plus screening: A3 plus qualified identity proofing at eIDAS level high

A grade does not discharge a relying party's customer due diligence (AMLR Art. 20): a relying party that is an obliged entity keeps its own obligations and records. The AML labels of A3 and A4 in Table 6 are those the issuer policy serves in production and on the public testnet. The validity is counted from the approval of the identification: a credential expires when the identification behind it does, and the way back is a new identification. A3 needs a recorded clear sanctions and politically-exposed-person screening and a second independent evidence; short of that the session grounds at A2. Freshness bands F1 to F4 are published by the issuer beside the grade and read on the classical seller rail; they are not a presentation predicate.

On production the identity-verification provider reports no due-diligence evidence, so every approval grounds at A2 and A3 is not issued on production; the public testnet's KYC sandbox (live) grounds A3 and verifies nobody. Every example in this paper that describes production uses A2.

5.4 The routes today

Today an accepted issuer verifies a person through an identity-verification provider, with a document and liveness check; this route is in production and grounds A2. The person identification data of a European Digital Identity Wallet, as a second evidence route at the issuer: the issuer's driver is built and switched off in every deployment; the relying-party side, the public testnet release and production are decided, not specified. On that route the issuer grounds A2 on the wallet's data alone, A4 only with a recorded clear screening and a second evidence independent of the wallet's provider, a rule that a contrary answer of counsel closes, and never A3. A person identified on that route enrols once: the identity credential it grounds is the principal from which the issuer mints the person's agents, by the same rules as on the other route, and an agent never outlives the wallet data's expiry or carries a grade above the person's; this path is built and switched off in every deployment, and not yet exercised end to end. The wallet data's status is read when it is presented; whether the issuer reads it again during the credential's life is open, and until it is decided a revocation of the wallet data reaches the person's credentials and agents at their expiry. An agent minted straight from the wallet's data, with no identity credential, is not specified. We are building KYB; it is not in production.

Only two routes exist, one in production. KYC, in this paper, names the regulated process an issuer runs and the route in production; it is not what the protocol requires.

5.5 What a verifier can require

A verifier can require a set of accepted issuers, a minimum issuer state and a grade threshold; an empty issuer set accepts any issuer active on the record layer. It cannot require a route, a provider, a freshness band or a date of

identification in a presentation. A verifier decides which accepted issuers it relies on and at what grade, never which route; each issuer publishes which routes it runs and how each grounds a grade, and the verifier relies on that policy being true.

An issuer that runs two routes issues credentials that a verifier cannot tell apart, so a verifier cannot accept one route of an issuer and refuse its other. A route shown in a presentation would split each issuer's holders into smaller anonymity sets, which is the privacy cost of a route a verifier could admit (§10.1).

06

The record layer

> A small set of public facts that no party should have to take from another's server.

6.1 The role

An issuer, a holder and a verifier rely on a small set of public facts that none of them should have to take from another's server: which issuers and verifiers are accepted, what each issuer's current status commitment is, and the one-time codes with which holders have switched their own credentials off. The protocol reads these facts from one shared record, the record layer; credentials stay with their holders, a proof goes from the holder to the verifier and nowhere else, and the verifier only reads the record layer to decide.

The record layer is not a passive log: it checks Ed25519 signatures on status commitments, emergency commitments and the Foundation's admission grants, a SHA-256 commitment over each batch of status updates, and the self-revocation proof, and it judges status commitments and one-time codes together, in one state. The reason is the one given in §3.8: two rules must not depend on a single party's goodwill, that only an issuer's key moves its status and that an issuer acknowledges a recorded code, and the record layer enforces both where every party reads, by rules every reader can check. That a code is recorded at all rests today on the single validator, and that the credential is removed rests on the issuer (§3.8).

6.2 What the protocol needs from the record layer

Table 7 states what the protocol needs from the record layer, and why.

Table 7 · What the protocol needs from the record layer.

WHAT THE PROTOCOL NEEDS FROM THE RECORD LAYER	WHY
Registry records written only when the MintID Foundation's admission, under its published policy, allows them, and a way for a reader to tell that a record carries that admission.	A verifier relies on an issuer because it was admitted, and a holder presents only to a verifier admitted under the published policy.
Each issuer's status commitments accepted only under that issuer's registered key, in a single order, each naming the one before it.	A holder's proof and a verifier's check must refer to the same status, and only the issuer may change it.
A block time that only moves forward and stays within seconds of real time.	The record layer refuses a status commitment that has expired, is too old or is dated more than a few seconds ahead of its block time, and a holder's revocation counts from the block time at which it is recorded; a block time running ahead of real time would lengthen the revocation bounds by as much.
A new status commitment from each issuer at least every 30 seconds, recorded and final within seconds.	A verifier refuses every credential of an issuer whose commitment has lapsed, so the issuer's next commitment must be final before the current one expires.
An issuer's emergency commitment recorded in the next block.	It is the shortest way to end a credential's authority.
One-time codes accepted from whoever carries them, on a proof that only a party holding the credential can make, checked by the record layer itself, with no account for the holder.	A holder can switch a credential off without the issuer's cooperation and without revealing who they are or which credential it is.

WHAT THE PROTOCOL NEEDS FROM THE RECORD LAYER	WHY
Status commitments and one-time codes judged together, in one state.	An issuer's commitment is refused if it does not acknowledge a code recorded long enough before it, so the issuer cannot ignore the code; the removal that follows rests on the issuer (§3.8).
Every record readable with a proof, by anyone, at no charge, at the moment of a decision.	A verifier checks the latest final state before it accepts and refuses when it cannot, and the protocol charges nothing to verify.
Records that grow with issuers, verifiers, validators and the protocol's own records (recorded governance decisions, self-revocations within their retention window, dispute claims, issuer-fault entries, issuers' key histories and, under the specification, disclosure receipts and registered arbiters), never with people or presentations.	No identification data of a holder is written, and the record stays small whatever the number of people.
Rules applied the same way by everyone who checks the record, with no outside input.	A record is valid because of the rules it passed, not because of who served it.
Governed values, such as how long a status commitment stays valid, changed only by a recorded decision that states what it changes.	These values set how long a person answers for an agent after a revocation, including for an access token issued on an acceptance before it (§4.3).
Records of the dispute rail and of disclosures: the commitment of each dispute claim and the receipts of its resolution, a receipt for every disclosure that never carries its content, and a registry of arbiters (the record of a claim and of a share's release in production; the rest specified, not built).	The MintID Foundation adds its share of the dispute key only for a claim recorded there, and every disclosure can be audited after the fact without its content.

Bonds, fee accounting and the compliance council belong to the chain's design and to the protocol's institutions; the protocol's guarantees between issuer, holder and verifier do not read them on a decision path, and the companion describes them. The chain's continuity is stated in §6.4, because "final once recorded" depends on it.

6.3 Time and the revocation bounds

Every way of ending a credential's authority has a start event and a bound after which every conforming verifier refuses the credential; a session or an access token that a transport built on a presentation accepted before the bound is part of that acceptance and can outlast the bound (§4.3). Table 8 gives the four paths with the symbols of Appendix B (T_H the heartbeat, T_A the maximum accepted age of a commitment, T_K the maximum future skew, T_G the grace for acknowledging a one-time code) and their values on the single-node production chain.

Table 8 · Revocation bounds.

PATH	START EVENT	EFFECT INSTANT	SINGLE-NODE PRODUCTION CHAIN
Emergency commitment	its finalisation	at finalisation	one block
Issuer status revocation	removal in the issuer's registry	start + T_A + T_K	185 s
Holder self-revocation, per-agent kill switch	the recording of the code	start + T_G + T_A + T_K	245 s
Principal-wide cascade	the trigger commitment	start + 7·T_H + T_A + T_K	395 s

A revocation takes effect at the bound of its path: one block on the emergency path, otherwise the stated bound from the start event at the deployment's governed values (185 s issuer, 245 s holder, 395 s cascade on the single-node production chain); the principal answers for what a conforming verifier accepts until then. These bounds follow from the record layer's ingestion rules, judged against its block time, and from the verifier's acceptance rules, judged against the verifier's own clock; they hold as stated while that block time does not run ahead of real time, an entry is final once it is recorded, the issuer removes what each path requires it to remove and, on the cascade, builds a commitment every heartbeat; on the holder path the record layer enforces the acknowledgement, not the removal (§3.8), and the cascade is issuer policy (§4.8). Under the specification the holder path's effect instant is its bound whether or not the commitment that acknowledges the code removes the credential: an acceptance after it that rests on a commitment that acknowledged the code without removing the credential is the issuer's fault, and the holder, for an agent its principal, does not answer for it. Likewise under the specification, on the principal-wide cascade, an acceptance after the effect instant that rests on a commitment without the agent's removal is the issuer's fault, whether the issuer removed the agent after the last of the eight commitments that follow the trigger or did not build a commitment every heartbeat, and the principal does not answer for it; no record of the record layer shows that a commitment kept such an agent, and the allocation holds without one. If an issuer's next status commitment is not final before its current one expires, no bound lengthens: every verifier refuses that issuer's credentials until a newer commitment is final.

Only the emergency path takes effect within one block. The issuer path starts at the issuer's own removal, off the record layer. The time before a code is recorded (relay delay, inclusion) is still the holder's to answer for. On the single-node production chain these bounds rest on one validator and one issuer, operated by one company.

6.4 MintID's chain as the record layer today

Today MintID runs on a single-node chain; running it on several nodes is foreseen as the project evolves. The chain holds the public records that make trust checkable: the registries of accepted issuers and verifiers, the issuers' status commitments, and the one-time codes with which holders switch their own credentials off. No identification data of a holder, credential, agent record or presentation is ever written to it.

The chain is a Cosmos SDK application chain on CometBFT, with single-slot deterministic finality. The protocol reuses that framework for consensus, networking and state proofs (ICS-23), and owns only the parts that are specific to identity: the registries, the status commitments, the acknowledgement rule and the verifier of the self-revocation proof. A verifier reads the chain from a node it runs or from a node of the operator that it designates, and proves every value against that node's state commitment; no light client is used. A third-party verifier can run its own node beside its service on the public testnet. On the single-node production chain the peer entry such a node would sync

from is built in the deployment code and not applied; once applied, it opens to the nodes of admitted integrators only. Until then every verifier there reads from a node the operator runs.

The chain's rules require the Foundation's signed admission grant on both verifier tiers: when a verifier registers itself or renews its origin, when a verifier admitted on request is registered, by a governance proposal voted by the chain's single validator, and on every other act of the chain's authority that changes a verifier's record, each grant used once; every verifier record of a genesis carries one too. The chain checks each grant against the admission key set, a governed value that the chain's governance installs and changes with no statement of the Foundation, so a grant shows the Foundation's admission only as far as that set is the Foundation's. The admission of an issuer carries no grant: it is decided by the Foundation and executed by the operator as a governance proposal voted by the single validator. Every decision is recorded in the admissions log, marked where it concerns a related party (§9.3). The compliance council is built into the chain and not seated. A verifier's record carries the issuer that identified its responsible person and that issuer's commitment; the chain checks the commitment's size only, and at a self-renewal it refuses a verifier whose issuer is no longer active. A verifier's origins are exact HTTPS origins, never a wildcard; for each, the chain keeps a digest of the verifier's evidence of control of the domain and the date until which that evidence holds, and an origin past that date fails step 1 of §3.6 until it is renewed.

Admission grant (both tiers; a signature, not a proof).

1. Sig_{ak}(a domain-separated statement of the act on the verifier's record and of the grant's expiry), under an active key $ak \in AK_{\mathrm{F}}$, checked by the record layer at every registration and renewal of a verifier and at every act of the chain's authority that changes a verifier's record.

On the single-node production chain the properties of Table 7 rest on two things, the published, signed software that applies the rules and the one node that the operator runs, with two exceptions: a reader can tell that a verifier's record carries the Foundation's admission only as far as the admission key set, which the chain's governance installs, is the Foundation's, and an issuer's record carries no grant; and of the records of the dispute rail and of disclosures, only the record of a claim and of a share's release is in production. A public blockchain operated by others could meet the same requirements; that option is studied, not built and not decided.

A verifier takes the state commitment from the node it designates without checking the validator's commit signatures itself. When that node is its own, the node has checked the validator's signature on every block it holds and has executed every block since the genesis, or since the snapshot it started from; a node started from a snapshot accepts the state at that height on the validator's signature. Every property that concerns who decides what is written, in which order and when rests on one validator key run by the operator, and no node can prove it: the order of transactions, the absence or delay of a transaction, a self-revocation included, the time written in a block and the chain's liveness. Every property that concerns what a write must satisfy is enforced by the published, signed software the node runs and is checked by a node that replays from the published genesis. Block history keeps every transaction, the self-revocation ciphertexts included; no block is pruned today, and the chain's data-retention policy is open.

Regeneration. Under the specification's rule for the chain's continuity, while every participant of the chain is a related party and no third party depends on its history, the chain's state may be replaced by a clean regeneration: a new genesis for which every signer is regenerated and nothing of the earlier chain is carried, no registration, issuer key, credential definition or recorded code. While the chain runs on a single node its operator can carry out such a regeneration under that rule. It has done so four times since 20 September 2026, on 24 September, 26 September, 4 November and 22 November 2026, so five chains have run, four retired and one in force. The retired chains are kept as test data: the operator keeps an archive of each with its genesis hash where it was recorded and the dates it ran, published with the chain-continuity pages of the integrator documentation, and no external anchor attests them today. No third party depends on the chain's state, so a further regeneration is clean; the custodian comes with a later clean regeneration, which follows the custodian's key ceremony and is not dated (§4.6). The operator commits to announce any further clean regeneration at least 14 days ahead, except one that the chain's security requires at once, which it announces when it makes it; this is its commitment, not a rule of the specification. From the first dependence of a third party on the chain's state, on the date the operator records, a new genesis carries that state forward as the chain's next generation. "Final once recorded" therefore holds within one chain: a credential issued before a regeneration does not verify after it, and its holder needs a new credential. A node that holds the chain notices a regeneration that happens after it joined; it cannot see one that happened before, and an external witness of the chain's hashes is not built.

07

The disclosure channel

> Identity detail never travels in a presentation; when it leaves, it leaves the issuer, on a path with its own trigger, recipient and status.

7.1 What the channel is

A presentation proves that an accountable, identity-verified person stands behind an agent, within a scope that person fixed, and reveals nothing about who that person is. Some relationships need more: a counterparty that a rule obliges to record its client, a business that wants to open an account, a party that an agent has harmed. The disclosure channel is the set of paths along which identity detail may leave the system once a credential exists, each with its own trigger, its own recipient and its own status. Identity detail never travels in a presentation and is never written to the chain; when it leaves, it leaves the issuer, the only party in the protocol that holds a record linking an agent to its person. Whoever runs the agent, or enrols it for the person, may know that link outside the protocol, and the issuer, its identity-verification provider and whoever holds a part of the dispute key remain subject to the law that applies to them (Invariant 5).

The paths are kept apart (Figure 6, Table 9). Where identity is wanted for a relationship, the person decides and refusal is the default; where it is wanted for accountability, due process decides and the person cannot veto it. No payment is ever the lawful basis of a disclosure.

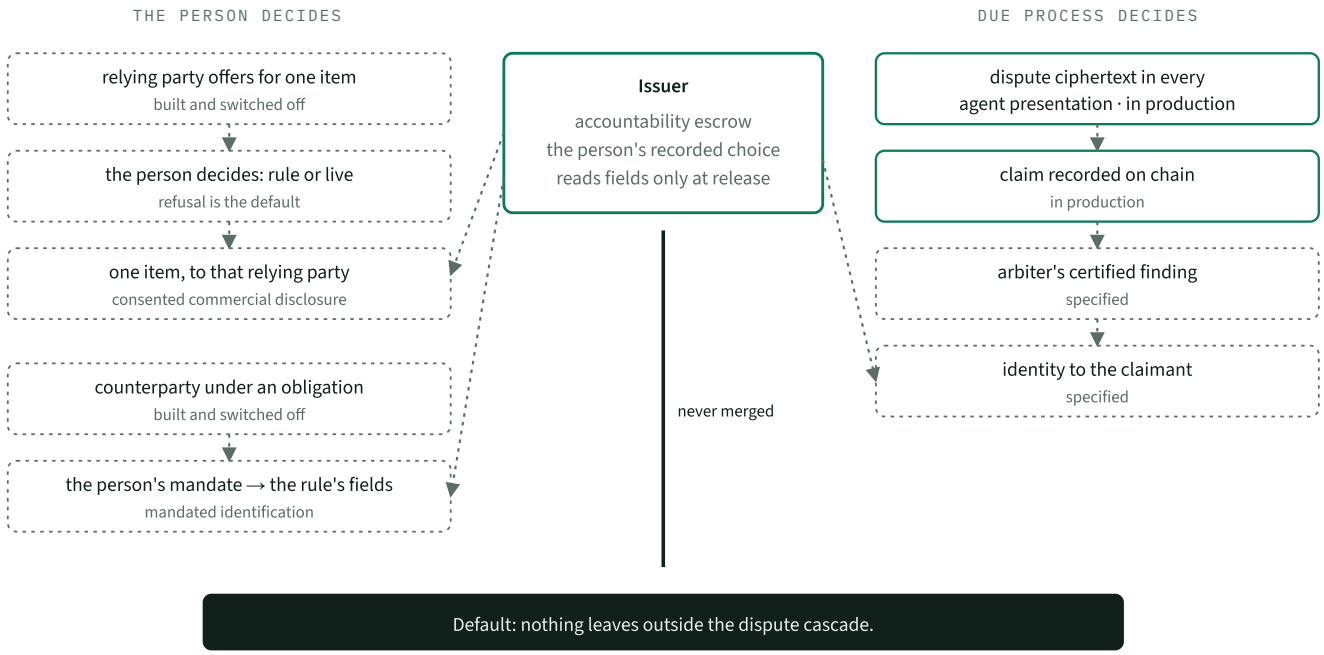

Figure 6 · Every path starts at the issuer; the person decides on one side, due process on the other. Solid: in production. Dotted: built and switched off, or specified, as marked.

7.2 No disclosure, the default

An agent credential carries no disclosure unless the person chose one when it was issued, and every agent credential issued today carries none. Under the specification the person chooses at issuance; as built, the issuer records the choice that the enrolling platform (the party that requests the agent's mint for the person, an operator of the commercial layer) submits with the mint request, and the person is not shown it before the mint. The choice is fixed

at issuance; under the specification it cannot be silently changed afterwards, widening it means a new credential, and narrowing it back costs the person nothing. “No disclosure” means no disclosure outside the dispute cascade described next, whose ciphertext every agent presentation carries and no person, issuer or relying party can switch off for an agent. The default is in production.

7.3 Identity under an arbiter’s certified finding

Every agent presentation to a registered verifier carries the dispute ciphertext: an encryption of the agent’s status reference under the dispute key, whose two parts are reserved to its issuer and to the MintID Foundation (custody: §4.6), with a proof inside the presentation that it encrypts the reference of the credential presented. Neither can open it alone, and under the specification the Foundation adds its decryption share, as a ministerial act, only for a claim recorded on chain. The ciphertext is fresh for every presentation, so it links no two presentations, and the verifier cannot read it; it keeps it, beside its decision record, for the claim window, longer only while a claim on it is open, and then deletes it. The dispute ciphertext is in production; it is new cryptography, and its audit scope is written; the audit is not yet commissioned. Each claim is recorded on chain as a hiding commitment, filed and signed by the verifier that accepted the disputed presentation (the filing released in the verifier library), and the chain records the receipt of each release of the Foundation’s share; under the specification the service that releases the share publishes that receipt in its own log and on chain before the share reaches the issuer, which alone completes the decryption; one share answers one ciphertext. That service, which reads each claim from the chain with its proof, is in the tree of the release and run by no deployment, so no share has been released; the resolution of a claim and the rest of the cascade are specified and not in production. Under the specification that service runs in infrastructure the operator runs until a later clean regeneration, which follows the custodian’s key ceremony, places it with a custodian.

Under the specification, when a disputed presentation was bound to a payment (§4.3), the verifier that files the claim may show the arbiter, and no other party, the payment’s terms as it kept them, which the arbiter checks against the digest inside the presentation’s request. That shows which payment the accepted presentation answered for. Only the verifier files a claim on this rail; a claim filed by the buyer’s side is parked, and the record of the payment’s terms that the agent’s side keeps, signed with the verifier’s request key, serves the principal outside the rail: with the seller, a payment company or a court. A claim recorded on chain lets the issuer, with the Foundation’s share, find the agent and notify its person, who may settle privately; a contested or unanswered claim goes to an arbiter that decides whether it is substantiated and sets an offer; a substantiated offer left unpaid leads to a minimal fraud signal, which names no one and carries coarse or banded attributes checked for cohort size, and then to the release of the person’s identity to the claimant, under the arbiter’s certified finding, for the claimant to pursue in court; the identity is also released when the claimant rejects the offer and the arbiter decides that the case belongs before a court. Identity is never released on payment, on a refusal to pay a claim found unsubstantiated, or on the claimant’s word alone, and no step asks the person’s consent. Under the specification the arbiter decides and certifies, the issuer releases, and the compliance council, once seated, supervises after the fact; the evidence bar of the arbiter’s finding and the conditions of escalation are a counsel item, and the identity release is not activated before that item closes.

Together the two key holders could open any ciphertext, so what keeps an opening inside a recorded claim is the separation of their custody and the claim as the only trigger; today that separation is procedural and rests on one operator and one person; from a later clean regeneration, which follows the custodian’s key ceremony, the specification places the Foundation’s part with a custodian that operates no issuer, chosen by the MintID Foundation, a related party of the operator (§4.6, §9.3).

7.4 Consented commercial disclosure

A relying party that wants a relationship with an agent's person (an account, a subscription, its own customer checks) may offer to learn one named item, and the person decides. The agent's pseudonym at that relying party is already in every agent presentation and lets it recognise the agent while the credential lasts, so the first option is to disclose nothing. Under the specification the items rise in identifiability: a single contact attribute (an email address, a telephone number or a postal address), then the person's identity (given name, family name, date of birth and nationality, without the number or the type of an identity document); a rule the person set at issuance may accept a contact attribute only for relying parties on a list the person set (by identity, accreditation or membership), within a cap and an expiry; the person's identity always needs the person's own decision at that moment. These items are built and switched off with the rail. A timeout, an error or the absence of a rule is a refusal; refusing costs the person nothing, and the relying party must keep its core service available without any disclosure. The agent never consents for its person. Under the specification an offer may carry a disclosure bid: a payment from the relying party to the person, made on acceptance, save as the next paragraph says of a release that is not delivered or that the relying party refuses, and taken by no one on refusal, which prices the person's consent and is never the lawful basis of the disclosure; the specification also sets a protocol cut on the bid and allows the issuer a brokerage fee, which the companion, *Governance, Economics and Launch Plan*, describes. As built and switched off with the rail, an offer carries its bid, a rule of the person may refuse a bid below a floor the person set, and the issuer's software reports the bid as payable only for a release delivered and accepted, and records no bid in the release's receipt. The settlement of a bid, its protocol cut and a brokerage fee are not built. Only the item asked for is released, only to the relying party that asked, and every release leaves a receipt that records that it happened and never what it contained. A disclosure gives no lasting right: to learn the item again after the credential is renewed, the relying party asks again.

The person decides on the principal control surface (§4.8), built and switched off in every deployment. Consented commercial disclosure is built and switched off; its activation waits for the consent governance texts, which are counsel items, and for the published demand criterion. The channel that delivers a release to the relying party is specified: under the specification the issuer delivers each release as an assertion signed under a key of its record on the record layer, to the origin at which the request's presentation was made, provided the relying party's record still registers that origin when the issuer delivers, bound to the request it answers, and receipts it. A release is delivered once the issuer has written its whole body towards the relying party's origin; a failure before or while it is sent leaves it not delivered. A release that is not delivered is consumed, makes no bid payable and is receipted as not delivered, and the issuer does not retry it; a release whose body was written and that the relying party answers as not accepted, answers outside the delivery contract or does not answer within the issuer's bounded time has left the issuer, is receipted as delivered and refused by the relying party, the receipt telling these cases apart, and its receipt records it as a disclosure; it makes no bid payable. The issuer's delivery, with its directory of relying parties read from the record layer, is built and switched off with the rail, and the receiving side at the relying party is built and switched off; no release can yet complete in a deployment.

Correlation is intrinsic to disclosure: unlinkability is not claimed for anything this rail releases, and a release of the person's identity reveals the exact grade and the source of the identification. Under the specification a release of a contact attribute carries neither; no release of a contact attribute can yet be made, since no identity-verification provider serves one, and a release of a contact attribute without the grade and the source is built and switched off with the rail.

7.5 Mandated identification

Some counterparties are obliged by a rule to record who their client is for the act itself: registering a guest, identifying a passenger, renting a vehicle. The person can authorise its agent once, with bounds, and the agent then performs the act without the person present. The fields are those the rule names for that kind of obligation, in a

governed list (in production, governed content of the record layer, whose genesis carries three classes as drafts, none in force, their rule citations awaiting counsel, a class once withdrawn staying withdrawn and a new obligation taking a new class; as built and switched off with the rail, the issuer serves only the classes the record layer governs, and none when it cannot read them, so the issuer in production serves none), never chosen by the counterparty; the mandate names the counterparty, or the obligation class whose registered counterparties it covers, carries a cap per period and an expiry no later than the credential's, and the person can revoke it at any time at no cost. The first identification towards each counterparty is the person's own decision; later ones within the bounds run without the person. The issuer reads the fields from its identity-verification provider at the moment of release, keeps none of them, and delivers to the counterparty a signed assertion, in the verified-claims form of OpenID Connect for Identity Assurance, that carries the grade and the source. Age is never a field; it is proven by predicate. The date of birth is a field only where the rule requires the date itself. This is the one path on which a relying party may make identification a condition of the act: if the identification does not come, it may decline the act and may attach no other penalty. Mandated identification is built and switched off, behind a switch of its own; the governed list of obligations is in production on chain, and the issuer's read of it built and switched off with the rail; the channel that delivers a release to the counterparty is specified, with the issuer's delivery built and the counterparty's receiving side built and switched off, no release can yet complete in a deployment, and activation waits for counsel.

7.6 What an issuer sets

Under the specification, an issuer cannot widen the channel on its own. It turns consented commercial disclosure and mandated identification on or off in its deployment, both off by default, and records at issuance the choice, the rules and the mandates of the person; the default of refusal, the person's own decision for identity, the fields of each obligation, the cascade's due process and the dispute ciphertext in every presentation are fixed by the specification and by governance. What the issuer and the Foundation learn on these paths is stated in Table 1.

7.7 Beyond the protocol today

Two modes go beyond what is built. The first is disclosure under conditions the person sets at issuance: within the consent rules and mandates above it is those paths, built and switched off; the specification also commits the person's choice into the credential, so that a relying party can see which paths an agent admits, and that commitment is specified, not built; conditions that would release identity without the person's own decision, outside a legal obligation, are excluded by the specification. The second is disclosure to any relying party, which no requirement specifies and of which nothing is built. The specification today lets a rule release a contact attribute only to relying parties on a list the person set, releases identity outside the dispute cascade only on the person's own decision or under a mandate, forbids making a consented disclosure the price of a service, and gives no relying party a lasting right to recognise the person. Neither mode beyond those bounds is claimed here.

Table 9 · The disclosure channel.

MODE	WHO SETS IT	WHAT IS REVEALED, AND TO WHOM	STATUS
No disclosure	The default; the person keeps it by choosing nothing	Nothing, to no one, outside the dispute cascade	In production
Identity under an arbiter's certified finding	No one per agent: every agent presentation carries the dispute ciphertext; an arbiter decides, the issuer releases	A notice to the person; on an unpaid substantiated offer, a minimal fraud signal that names no one and then the person's identity, to the claimant; the identity also when the claimant rejects the offer and the arbiter refers the case to a court	The dispute ciphertext in production; the cascade specified
Consented commercial disclosure	The person, under the specification, by a rule set at issuance or a decision at the moment (as built, the rules arrive with the enrolling platform's mint request, §7.2); the relying party offers	One named item, a contact attribute or the person's identity, to the relying party that offered	Built and switched off; its items as specified built (§7.4)
Mandated identification	The person's mandate at issuance; the obligation and its fields by governance; the counterparty asks	The fields the obligation names, to a counterparty the mandate nominates, by name or by obligation class	Built and switched off
Disclosure under conditions set at issuance	The person, at issuance	Within the two paths above, as those; beyond them, nothing	As consented commercial disclosure and mandated identification (built and switched off); the commitment in the credential specified
Disclosure to any relying party	Not specified	Not specified	Not specified: no requirement specifies it and nothing of it is built

08

Security and privacy properties

> Thirteen claims, each with the party it protects, its assumptions, a test and its limit.

8.1 How each property is stated

Each property is stated as a claim, followed by: the party it protects and against whom; the assumptions it rests on, named as in Table 3; a test a reviewer can apply to the specification or to a deployment; its status; and its limit, which is part of the claim. The properties are this paper's own statement of what the construction gives. None of them is yet backed by an independent audit (§11). Table 11 (§8.15) lists the thirteen properties at a glance.

"In production", for a property, means that its mechanism runs on the single-node production chain. It does not mean that the property is exercised there between distinct parties: on that chain one operator runs the issuer, the validator, the privacy relay and the only admitted verifier, ZadQ's production clients use the classical seller rail, and no third-party verifier is admitted (§9.3). A property whose claim is against a distinct party (P3, P4, P9, P10 and, against the issuer, P11) holds today only where the parties are in fact distinct, and each cites the operator assumption T-O.

8.2 P1. Accountability

Claim. A verifier that accepts a presentation knows, from the presentation and the record layer alone, that the proof was made for its own challenge; that it was made with a credential an accepted issuer signed, which is not revoked at the verifier's latest read and unexpired at the reference instant, which every policy asks; that the credential's grade is at least the one asked; and, for an agent, that the issuer signed the credential under its agent profile, which it does only for an agent it minted from a person's identity credential, together with the agent's pseudonym for that verifier and a ciphertext through which a recorded claim can reach the agent.

Protects. The verifier, against a holder or an agent that would present without an accountable person behind it, with another's proof, or with a revoked or expired credential.

Assumptions. q-SDH, and AGM for blind issuance; DL; ROM; Ed25519 and SHA-256, for the status commitment and the inclusion of the accumulator and the definition under it; T-I, for what the issuer signs and when it signs under the agent profile; T-P, for the grade the issuer grounds; T-R, for current, authenticated state.

Test. Present with a credential of an issuer outside the verifier's accepted set: refused. Replay or rebind the proof: refused. Present an agent credential without the pseudonym or the dispute ciphertext: refused. Revoke and present after the effect instant: refused.

Status and limit. In production, the binding of the mint presentation's disclosed reference to the credential it proves, on which the agent clause rests, included (§4.2). What an agent credential says of its person is the issuer's attestation at mint, not a proof (§4.2), and no agent presentation checks the person's credential again: its revocation reaches the agent through the issuer's cascade, within its bound (§4.8, §6.3); the custody rank is the issuer's appraisal, and whoever holds the link secret presents (§4.3). A verifier relies on issuers and a grade threshold, never on an identification route (§5.5). A credential can be accepted up to the presentation lifetime after its expiry instant (§3.4).

8.3 P2. Continuity

Claim. Within one verifier, every presentation of one agent credential carries the same pairwise pseudonym, proven equal to the link secret the issuer signed, so that the verifier recognises the same agent credential across its sessions

for the credential's life.

Protects. The verifier's ability to hold one agent to its own record (its history, its limits, a refusal), without learning who stands behind it.

Assumptions. q-SDH, for the binding of the pseudonym to the signed link secret, and AGM for blind issuance; DL; ROM; T-I, for what the issuer signs; and the verifier deriving the pseudonym generator from its own identifier, which its acceptance rules do.

Test. Two presentations of one agent credential to one verifier carry equal pseudonyms; to two verifiers, unequal ones; a re-minted agent carries a new one.

Status and limit. In production. Continuity is of the agent credential, not of the person: one person may hold several agents within the issuer's cap, each with its own pseudonym, and a re-mint resets it; a per-person, per-verifier tag that would hold one person to one limit across that person's agents is open. There is no continuity across verifiers, by design (P9), and none along a chain of agents: delegation from one agent to another is deferred. Identity presentations carry no pseudonym.

8.4 P3. Non-identification

Claim. From a presentation a verifier learns the accepted issuer, the profile, the result of each predicate it asked for, and, for an agent, the scope values it named and the agent's pseudonym for it. It learns neither the grade, any date, the limit, the unit, the custody rank, the status reference nor the link secret, and no party other than the issuer can obtain from presentations an attribute that identifies the person. Two presentations of one credential to two verifiers carry no common value other than the issuer, the profile, the accumulator of the epoch they were proven against and public parameters.

Protects. The holder and the person, against verifiers, alone or together.

Assumptions. ROM, for the zero knowledge of the composed proofs; the perfect hiding of Pedersen commitments, which requires fresh randomness from the holder's prover (T-H); DDH in G_1, for the pseudonym and the dispute ciphertext; T-F, for one Foundation key record per version; T-O, where one operator runs the issuer and a verifier.

Test. Structural: list every value that two presentations of one credential to two verifiers have in common; the list is the one above.

Status and limit. In production; between the issuer and a verifier it holds only where they are distinct, and on production the issuer's operator runs the only admitted verifier (§8.1). The anonymity set is the issuer's holders of that profile, and on production there is one issuer. The accumulator a presentation names reveals roughly the age of the holder's witness. A verifier that asks different thresholds or reference instants across sessions learns more than one presentation reveals, and the holder library has no rule against it; an agent's sessions at one verifier are linked by its pseudonym. A narrow issuer allow-list can reveal the identification route by inference (§5.1). The classical seller rail reveals the grade and the freshness band, and a release on the disclosure channel reveals what it releases. Network, timing and payment metadata are outside the proof. A verifier cannot detect a holder prover that does not draw fresh randomness (§9.2, item 17).

8.5 P4. Conditioned resolution

Claim. Within the protocol, only the issuer can connect an agent to its person, through its accountability escrow (parties outside the protocol: §9.2 item 20), and every access to the accountability escrow leaves an entry in a hash-chained trail. A party holding an agent presentation can reach the agent only through the dispute ciphertext the

presentation carries, which opens with a decryption share of the issuer and one of the MintID Foundation, each with a proof bound to the claim and to that presentation, and never with one of them alone. Under the specification, the Foundation adds its share only for a claim recorded on chain, and the issuer releases the person's identity only under an arbiter's certified finding.

Protects. The person, against de-anonymisation by any single party and against de-anonymisation outside due process; the claimant's ability to reach an agent that harmed it.

Assumptions. DDH in G_1, for IND-CPA; DL and ROM, for the proofs of possession and the decryption-share proofs; Ed25519, for the Foundation's key record; HMAC-SHA-256 as a pseudorandom function, for the derivation of the Foundation's part; T-H, for a dispute conjunct made by the published prover and for the holder's check of both proofs of possession; T-F and T-I, for the custody of each part; T-O, for the separation of the two custodies.

Test. Open a ciphertext with one share: what remains is a ciphertext under the other part. Lift a ciphertext into another presentation: refused. Present a share for another claim or presentation: its proof fails. Replace one part by a function of the other: its proof of possession fails.

Status and limit. The dispute ciphertext, the retention record, the claim record on the record layer, the accountability escrow's trail and the binding of the mint presentation's disclosed reference to the credential it proves, from which the escrow's link to the person is recorded (§4.2), are in production; the share release after a recorded claim is specified, in the tree of the release and run by no deployment, and the arbiter registry and the identity release are specified (§7.3). The two-party property is cryptographic and the separation of custody procedural: today both parts are kept in infrastructure the operator runs, and one person holds the credentials of both roles; a custodian that operates no issuer, chosen by the MintID Foundation, a related party of the operator, is specified for a later clean regeneration, which follows the custodian's key ceremony (§4.6, §7.3). That the issuer, which would receive the Foundation's share for each recorded claim, learns nothing from those shares about other ciphertexts is an open step of the project's draft argument, which has not been reviewed outside the project (§4.6). No service that releases the Foundation's share runs in any deployment, so the Foundation's precondition, a claim recorded on chain, is a rule of the specification and of that service's code, not one a deployment exercises (§7.3). The accountability escrow's trail is the issuer's own: its entries are hash-chained and its heads signed and kept by the issuer; no head is committed to the record layer and no third party receives them today (the council that would is not seated), so the trail shows a rewrite only to someone who already holds an earlier head. The claim window and the data-protection framework of the retention record are as §4.6 states.

8.6 P5. Unobserved status

Claim. A verifier establishes that a credential is unrevoked without sending the issuer or any status provider a value that names the credential: the holder proves that its hidden status reference is a member of an accumulator covered by the issuer's current status commitment or by a recent one that §3.6 accepts, and the verifier reads those commitments, per issuer, from the record layer. The issuer does not see the proof, and nothing a verifier sends at a decision names the credential; a verifier that takes the issuer's published definition from the issuer when it decides, as the reference verifier service does, shows the issuer when it decides. A holder brings its status witness forward from the issuer's public status log, on a schedule common to the holders of its profile, and sends nothing that names its credential; a holder that asks the issuer for its witness instead (§3.7) names its credential, so the issuer learns when that credential is active and, joined with the time of the definition reads above, can learn where it presents.

Protects. The holder, against status providers and the node a verifier reads from, as to which credential is checked and where it presents; against the issuer, as far as the claim states.

Assumptions. q-SDH, for accumulator soundness; ROM, for zero knowledge; Ed25519 and SHA-256, for the commitment; T-R.

Test. The verifier's status read carries no per-credential value: it names the issuer and reads its commitment and accumulator. The holder's fetch of the public status log names no credential; a witness request to the issuer does, which the claim states.

Status and limit. In production. The server of the public status log learns which network addresses fetch it and when, and the log leaves the issuer's assertion of each removal unproven (§3.7); that its padding hides how many removals were due is pending audit (the privacy review of the public status log). The node a verifier reads from learns which issuer's commitment it reads, not which credential.

8.7 P6. Bounded withdrawal

Claim. Each way a credential's authority is withdrawn has a stated start event and a stated bound, after which every conforming verifier refuses it: the emergency commitment, at its finalisation (one block); issuer status revocation, from the issuer's removal, within $T_A + T_K$; holder self-revocation and the per-agent kill switch, from the recording of the one-time code, within $T_G + T_A + T_K$; the principal-wide cascade, from the trigger commitment, within $7 \cdot T_H + T_A + T_K$; the values on the single-node production chain are those of Table 8. On the holder path that refusal holds when the commitment that acknowledges the code also removes the credential, and on the cascade when the issuer removes each agent by the last of the eight commitments that follow the trigger and builds a commitment every heartbeat. Until the effect instant, the holder, for an agent its principal, answers for every presentation a conforming verifier accepts, and for a session or an access token that a transport builds on such an acceptance, which can outlast the effect instant (§4.3); who answers for an acceptance after the effect instant is a rule of the specification, stated under Status and limit, and not part of this claim. A verifier that cannot read and prove the current state refuses.

Protects. The holder and the person, with a bounded exposure after they act; the verifier, with a bounded time during which a withdrawn credential may still be accepted.

Assumptions. T-R: a block time that moves forward and does not run ahead of real time, and an entry final once recorded; the verifier's own clock (Clocks and cadence); T-I on every path: on the emergency and issuer paths the bound starts at the issuer's own commitment or removal, which must remove the credential; on the holder path the record layer enforces only that the issuer acknowledges the code, and the refusal at the bound holds only if the commitment that acknowledges it also removes the credential; on the cascade the issuer removes each agent by the last of the eight commitments that follow the trigger and builds a commitment every heartbeat, and nothing on the record layer shows it when it does not; a verifier that conforms by reading at decision. A late commitment lengthens no bound: verifiers refuse the issuer's credentials until a newer one is final, which is a matter of availability and not a condition of the bounds.

Test. Withdraw at t_0, measured from the stated event; a conforming verifier refuses at or before t_0 plus the bound. The revocation trace of the public testnet (live) measures it.

Status and limit. In production. Only the emergency path takes effect within one block. The limits of each path, an issuer that acknowledges a code without removing the credential among them, are those of §6.3 and §3.8. Under the specification the holder path's effect instant is its bound all the same, and an acceptance after it that rests on a commitment that acknowledged the code without removing the credential is the issuer's fault, not the holder's; on the principal-wide cascade, an acceptance after the effect instant that rests on a commitment without the agent's removal is the issuer's fault, not the principal's, whether the issuer removed the agent after the last of the eight commitments that follow the trigger or did not build a commitment every heartbeat. Both are rules of the

specification, not properties this section proves. The record layer cannot see such a commitment: under the specification a holder can prove one there only within the evidence window, and only against an issuer that left the credential out of its committed removals (§3.8); that evidence covers recorded self-revocation codes, not the removals of a cascade, so no record shows a commitment that kept a cascaded agent. The allocation to the issuer holds where no such evidence exists.

8.8 P7. Binding, single use and freshness

Claim. Every presentation proof is bound, as its Fiat–Shamir nonce, to a digest of the verifier's origin, its identifier, its request key, a fresh single-use nonce, the whole signed request and the finalised height the verifier read; it is valid for the presentation lifetime of §3.5; its nonce is consumed once, in the verifier's own durable store, after every other check. A presentation made for one verifier, one profile or one policy never verifies for another; the mint, relay and self-revocation proofs are bound to their own challenges and statements; a dispute ciphertext cannot be lifted into another presentation.

Protects. The verifier, against replay and against a proof made for another verifier, policy or challenge; the holder, against its proof being accepted at any verifier other than the one it answered.

Assumptions. ROM; SHA-256 collision resistance; Ed25519, for the challenge signature the holder checks; the verifier's clock and durable store (Clocks and cadence).

Test. Replay within the lifetime: refused. Answer at or after expiry: refused. Present to another verifier or under another policy: refused. Lift the dispute ciphertext: refused.

Status and limit. In production (§3.5). Replay protection is per verifier, the lifetime is measured on the verifier's clock, the binding to the status commitment is indirect, and verification is online (§3.5, §3.6). A real-time relay of a verifier's genuine challenge to that same verifier is stopped only by the holder's host (§3.5). On the x402 transport with Web Bot Auth, a presentation is accepted only in a request signed by the key its challenge was issued to (§4.3; released, in the x402 middleware and the agent clients; built and switched off in the verifier service in production).

8.9 P8. Soundness of credentials and predicates

Claim. A presentation that verifies proves knowledge of an issuer's signature over claims the holder cannot change, its link secret among them, and that each requested predicate holds for the signed value: grade at least γ_0; unexpired at the reference instant; over n' years; for an agent, key-custody rank at least κ_0, transaction limit at least λ_0 in the unit the policy names, and membership of its signed scope reference in each named value's set.

Protects. The verifier, against a holder that would claim a grade, an age, a limit or a scope it was not given.

Assumptions. q-SDH, for BBS and VB20, and AGM for blind issuance; DL, for Pedersen binding and Bulletproofs; ROM; Ed25519, SHA-256 and T-R, for the keys and accumulators read under the status commitment; T-I, for what the issuer signs; T-P, for the values the issuer signs.

Test. Structural: every predicate is a range statement on a commitment tied to the signed claim by a shared response, and an accepted presentation fails under any other bound, policy, nonce or accumulator, as the published vectors show. That no cheating prover can satisfy a false predicate is a soundness argument, the question of the scheduled implementation audit; no vector can show it.

Status and limit. In production, including, for the presentations made to the issuer (mint and relay), that each disclosed claim is the signed value (§4.2); the age floor of agents is built, not issued (§4.1). The values are the issuer's.

The soundness of the composition is the scheduled audit's question, and a presentation answers a policy with one credential (§3.4).

8.10 P9. Unlinkability, case by case

Claim. Unlinkability is not one property but several, one per pair of parties that might try to link. Table 10 states each case: what holds, under which assumption, and with which limit.

Table 10 · Unlinkability, case by case.

CASE	WHAT HOLDS	ASSUMPTION	LIMIT
Presentations of one identity credential, to one verifier or two	Not linkable through their contents	ROM; fresh prover randomness (T-H)	The accumulator reveals the witness epoch's age; side channels
An agent's presentations at one verifier	Linkable by design, through the pseudonym	none	none
An agent's presentations at two verifiers, alone or colluding	Not linkable through their contents: the pseudonyms differ and the dispute ciphertexts are fresh	DDH in G_1; ROM; T-H (fresh prover randomness; the holder refuses another verifier's pseudonym); T-F (one Foundation key record per version)	An audit item; an operator that runs several verifiers holds their records; no published holder host keeps the Foundation's key record (§9.2, item 22)
Two agents of one person, by the record layer, a verifier or two verifiers	Not linkable	Fresh link secrets by the holder (T-H), fresh references by the issuer (T-I), which no verifier can check	The issuer knows the link
A renewed or re-minted credential and the one it replaces	Not publicly linkable	As in the first row	The issuer links them
The issuer with a verifier	Cannot link a presentation to a credential through its bytes; the issuer cannot compute an agent's pseudonym	ROM for the hiding of the link secret, DDH in G_1 after a self-revocation; for an agent, DDH in G_1 under the Foundation's part, for the dispute ciphertext the issuer cannot open alone; T-O	The holder's fetch of the public status log names no credential, and the log's server and the node that serves the holder's reads still learn which network addresses fetch or read and when, and a witness request to the issuer names the credential (§3.7); on production the issuer's operator runs the only admitted verifier
An observer of the record layer	From a self-revocation: the issuer, the profile, the code, the witness epoch's age and the time; not the credential or the holder; the size of a person's set of agents is not countable from later commitments	DDH in G_1; ROM; HMAC-SHA-256 as a pseudorandom function for the issuer's cover removals and cascade draws in the public log	Cascade unlinkability is an audit item, and the log's padding pending audit (the privacy review of the public status log); a commitment whose removals exceed one batch shows that more removals than one batch were due at that commitment, and which commitments remove an identity credential is public

The ways presentations become linkable are listed once, in §9.2 item 1. A verifier that publishes its verdict outside the protocol, as the published adapter for an agent registry on a public chain does on the agent owner's request, links what it publishes; the cases of Table 10 cover what the protocol carries.

Protects. The holder and the person, against correlation.

Test. For each row, the structural test of P3 applied to the parties named.

Status. In production as mechanisms; every row whose parties production does not separate holds there only as far as T-O (§8.1); the implementation audit and the self-revocation audit are scheduled, and the dispute-ciphertext audit has its scope written.

8.11 P10. Issuer blindness to presentations

Claim. The issuer never receives a presentation made to a verifier, and it cannot compute an agent's pseudonym at any verifier. It learns when a credential refreshes its status witness from it, which a holder that updates from the public status log does not do, and, from a verifier that reads its published definition at each decision, when that verifier decides; joined, the two can show it to which verifier a credential presented. It learns which agent a pseudonym names when that agent relays a relying party's offer to it, or when a recorded claim is opened.

Protects. The holder and the person, against the issuer's sight of their presentations and their pseudonyms; its view of where they go is limited only as the claim states.

Assumptions. ROM, for the hiding of the link secret before a self-revocation, and DDH in G_1 after; zero knowledge; T-F, in that the issuer does not obtain the Foundation's share outside a claim; T-O, in that the issuer's operator does not join what it sees as an issuer with what it sees as a verifier.

Test. The issuer's inputs after issuance are fetches of its public status log, which name no credential, and any witness requests, which do; the presentations made to it, the principal's mint presentation and an agent's relay presentation, each of which discloses a status reference; and, for a recorded claim, the claim's ciphertext and the Foundation's share. No presentation made to a verifier reaches it outside a recorded claim; a verifier's reads of its published definition reach it with their time.

Status and limit. In production as a mechanism; on production the issuer's operator also runs the only admitted verifier, so the property holds only between distinct parties (§8.1). A witness request to the issuer, joined with any verifier's log or with the time of a verifier's definition reads, can link presentations to credentials; the public status log leaves its server the time and network address of each fetch (§3.7).

8.12 P11. Self-revocation without the issuer's cooperation

Claim. A holder that holds a status witness valid against a commitment in the issuer's ring (for an agent, one refreshed within the last eight commitments) can submit, without the issuer's cooperation, without an account on the record layer and without revealing to anyone but its issuer which credential it is, a self-revocation that the record layer verifies; once the record layer records its code, it refuses every status commitment of that issuer that has not acknowledged the code within the grace, so that the issuer must acknowledge the code or all its credentials stop verifying; that the acknowledging commitment also removes the credential rests on the issuer.

Protects. The holder and, for an agent, its person, against an issuer that would not revoke, and against being identified by revoking.

Assumptions. q-SDH; DL; AGM, for one degenerate case of the soundness argument; DDH in G_1; ROM, for the soundness and the witness indistinguishability of the conjunction; Ed25519 and SHA-256, for the keys, the accumulator and the encryption key read under the issuer's signed status commitment; T-I, for the removal that follows the acknowledgement; T-Rel; T-R, in that the record layer includes the submission; T-O, where one operator runs the relay and the issuer.

Test. Submit a valid self-revocation and withhold the issuer's acknowledgement: the issuer's next commitments are refused. Submit a proof for a code not signed into a held credential: refused, which is the defence against arbitrary revocation. Replay across chains or regenerations: refused.

Status and limit. In production; the self-revocation audit is scheduled, and the project's soundness and witness-indistinguishability arguments for the proof are written, not yet published and not reviewed. Self-revocation is itself a governed value of the record layer that a recorded decision can switch off, and on the single-node production chain that decision is the single validator's vote, which the operator casts. The other limits are those §3.8 lists: acknowledgement without removal, visible only to the holder, which the holder can record as the issuer's fault within the evidence window when the issuer left the credential out of its committed removals; the code's relation to the link secret, certified and not proven; a new composition and a re-implemented signature proof; a cached witness that lasts only the ring for an agent, after which the issuer can withhold the fresh one; the relay and the issuer under one operator; the validator's power to delay or censor; a code not unique to one credential, whose record the holder library, as released, counts as its own only by its own ciphertext, and whose record under another credential nothing yet prevents; and the unproven confirmation read. Every limit of this property is within the scope of the scheduled self-revocation audit.

8.13 P12. Scope containment

Claim. An agent cannot widen its scope: its scope reference and its limit are signed into its credential at mint and proven by membership and range, and a membership witness borrowed from another agent is refused.

Protects. The person, in that an agent acts only within what was delegated, and the verifier.

Assumptions. q-SDH; DL; ROM; Ed25519, SHA-256 and T-R, for the keys and accumulators read under the status commitment.

Test. Present an agent for a scope value it was not minted with: refused; with a witness of another agent: refused.

Status and limit. In production. The scope directory, the limit that counts nothing and the absence of a time window are as §4.4 states.

8.14 P13. Nothing holder-specific on the record layer

Claim. No record-layer state or message carries a holder's name, account, credential identifier, identification record, presentation or proof history; state grows with issuers, verifiers, validators and the protocol's own records (recorded governance decisions, self-revocations within their retention window, dispute claims, issuer-fault entries, issuers' key histories and, under the specification, disclosure receipts and registered arbiters), never with holders or presentations; issuing a credential writes nothing, and minting an agent writes only an aggregate count.

Protects. Holders and persons, against anyone who reads the record layer.

Assumptions. None beyond the schema of the record layer; T-R, for what nodes keep.

Test. Walk every field of every state type and message: none names a holder. A test of the chain does so on every build.

Status and limit. In production. The guard checks field names, not values. Self-revocations write public one-time codes and, until the evidence window after the issuer's fold ends, and longer while a fault on it is open, a ciphertext for the issuer; block history keeps every transaction until pruned, and the chain's data-retention policy is open (§6.4). A verifier's record names the verifier and its origins and carries an issuer's commitment about its responsible person, the one person-related value on the record layer.

8.15 The properties at a glance

Table 11 gathers the thirteen properties; each row's limit is the principal one, and the property's own section states the rest.

Table 11 · The properties at a glance.

PROPERTY	PROTECTS	ASSUMPTIONS	STATUS	PRINCIPAL LIMIT
P1 Accountability	Verifier	q-SDH, DL, ROM, AGM, Ed25519, SHA-256, T-I, T-P, T-R	In production	What an agent says of its person is the issuer's attestation at mint
P2 Continuity	Verifier	q-SDH, DL, ROM, AGM, T-I	In production	Of the agent credential, not of the person; reset by a re-mint
P3 Non-identification	Holder, person	ROM, T-H, DDH, T-F, T-O	In production	One issuer's holders as anonymity set; probing; metadata; issuer and only verifier one operator on production; the Foundation's key record kept by no published holder host
P4 Conditioned resolution	Person, claimant	DDH, DL, ROM, Ed25519, SHA-256, T-H, T-F, T-I, T-O	Ciphertext and claim record in production; resolution specified	Both parts in the operator's infrastructure, one person's credentials; no share-release service runs in any deployment
P5 Unobserved status	Holder	q-SDH, ROM, Ed25519, SHA-256, T-R	In production	The log's server sees which addresses fetch it; the issuer sees when verifiers read its definition
P6 Bounded withdrawal	Holder, person, verifier	T-R, clocks and cadence, T-I	In production	The issuer's removal; one validator and one issuer, one company
P7 Binding, single use, freshness	Verifier, holder	ROM, SHA-256, Ed25519, verifier clock	In production	Per verifier; the verifier's clock; real-time relay left to the host
P8 Soundness	Verifier	q-SDH, DL, ROM, AGM, Ed25519, SHA-256, T-I, T-P, T-R	In production	The values are the issuer's
P9 Unlinkability	Holder, person	ROM, DDH, T-H, T-I, T-F, T-O, SHA-256	In production	Status-log fetches by network address; issuer and only verifier one operator on production; the Foundation's key record kept by no published holder host
P10 Issuer blindness	Holder, person	ROM, DDH, T-F, T-O	In production	Status-log fetches by network address and verifiers' definition reads; issuer and only verifier one operator on production
P11 Self-revocation	Holder, person	q-SDH, DL, DDH, ROM, AGM, Ed25519, SHA-256, T-I, T-Rel, T-R, T-O	In production	Acknowledgement without removal is visible only to the holder; a governed switch turns it off
P12 Scope containment	Person, verifier	q-SDH, DL, ROM, Ed25519, SHA-256, T-R	In production	The scope directory; no time window
P13 Nothing holder-specific	Holders, persons	Schema; T-R	In production	Name guard, not a proof; retention open

09

Threat model and residuals

> What each adversary gets, what it does not, and what remains.

9.1 Adversaries

Table 12 takes each adversary in turn and states what the protocol gives against it and what it does not give. The second column refers to the properties of §8; the third to the residuals of §9.2, by number.

Table 12 · Adversaries.

ADVERSARY	WHAT THE PROTOCOL GIVES	WHAT IT DOES NOT GIVE
A curious verifier	It learns the issuer, the profile, the predicate results it asked for and, for an agent, the named scope values and the pseudonym, and nothing else (P3); it cannot reuse a proof (P7)	More across its own sessions by probing (2); an agent recognised across its sessions, by design (1)
Two verifiers, colluding	No link through contents: different pseudonyms, fresh ciphertexts (P9)	Metadata and published verdicts (1); an operator running both holds both records
A verifier relaying another verifier's challenge	Nothing beyond the holder's host (P7)	(15)
The issuer	It never sees a presentation made to a verifier, cannot compute pseudonyms and cannot open a dispute ciphertext alone (P10, P4)	Every claim it signs, every mint link, every code it decrypts, which addresses fetch its status log, any witness request and the timing of verifiers' definition reads (4); trust for what it attests of an agent's person (6)
The issuer with a verifier	No link of a presentation to a credential through its bytes	Status-log fetches or a witness request joined with the verifier's log (4); on production, one operator (8)
The issuer with the MintID Foundation	Nothing beyond the separation of their custody, procedural today (a custodian that operates no issuer, chosen by the MintID Foundation, a related party of the operator, is specified for a later clean regeneration, which follows the custodian's key ceremony)	Together they could open any dispute ciphertext a verifier hands over (11)
The privacy relay	It learns the code, the issuer, the profile, the epoch of the commitment the proof names (hence the witness's age), the time and a ciphertext it cannot read; it never holds an address with a code, at the application layer	What stands in front of the application; on production, the relay and the issuer under one operator (12)
An observer of the record layer	Registries, commitments and codes; no holder, credential or presentation (P13, P9)	The witness epoch's age and the timing of self-revocations; a public, permanent code (1)
A malicious holder	It cannot forge claims, widen its scope, revoke another's credential or replay (P8, P12, P11, P7), or show its issuer another credential's status reference at a mint or a relay (§4.2)	It can copy its link secret to another presenter (§3.9)
A node a verifier or a holder reads from	Values are proven against its state commitment	The commitment's authenticity is the designated node's (5); a holder's read of a verifier's record names that verifier

ADVERSARY	WHAT THE PROTOCOL GIVES	WHAT IT DOES NOT GIVE
The operator, as a related party standing behind several roles	Every rule that concerns what a write must satisfy is enforced by the published, signed software and can be checked by replay	It could withhold or order writes, change the admission key set or switch holder self-revocation off by its validator's vote, regenerate the chain, and stands behind both custodies of the dispute key (8 to 13)

9.2 Residuals

The residuals this paper states are listed here, each once, with the section where it is stated in full.

1. **Linkability cases.** Presentations are linkable through the pseudonym at one verifier; to the issuer when an agent relays or a claim is opened; through the public self-revocation code; through a holder prover that does not draw fresh randomness; through a verdict a verifier publishes outside the protocol; and through network, timing or payment metadata. The payment digest that the specification allows (§4.3) adds none of these: it is drawn fresh for each challenge, never published and never on a ledger. The payment's terms a verifier keeps beside it would locate a payment on a public ledger, which is why they reach only an arbiter, and only in a claim. The anonymity set is one issuer's holders of a profile. A re-mint resets the pseudonym, and one person may hold several agents within the cap (§8.4, §8.10).
2. **Predicate probing.** A verifier that varies thresholds or reference instants across linked sessions learns more; the holder library has no rule against it (§3.5, §8.4).
3. **Online verification, fail closed.** Every decision needs the record layer's live state, and a silent issuer takes its credentials offline within the maximum accepted age of its commitment (§3.6, §3.7, §6.3).
4. **The issuer sees status-log fetches and witness requests.** A holder updates its own witness from the issuer's public, padded status log without naming its credential, and the log's server learns which network addresses fetch it and when; a holder that asks the issuer for its witness names the credential, and the issuer can withhold that witness; a verifier that reads the issuer's published definition at each decision adds the time of that decision; a removal listed in the log stays the issuer's assertion, removal is not provable in the accumulator, and the hiding the padding gives is pending audit (§3.7).
5. **Reads from a designated node, not a light client;** a third party's own node cannot sync from the single-node production chain today (§6.4).
6. **Trust in the issuer for an agent's person.** Derivation, the grade bound, the age floor, the cap, the freshness of the identification, fresh references for a person's agents, the custody appraisal and the removal of a revoked person's agents, which no rule of the record layer enforces, are the issuer's. The cap counts the sessions the issuer links to one person, and on the wallet route nothing tells the issuer that two unlinked sessions are one person's (§4.2, §4.3, §4.8, §5.4).
7. **The grades an issuer may assert are self-enforced** (§5.2).
8. **One node, one operator.** The single-node chain, its single validator and its single operator; the revocation bounds rest on them (§6.3, §6.4).
9. **Admission by one validator's vote.** Issuer admission, the admitted verifier tier and the admission key set are executed as governance proposals voted by the single validator; the chain's rules require the Foundation's grant on both verifier tiers and on every act of the authority that changes a verifier's record, never for an issuer, and check it against the admission key set, so a grant shows the Foundation's admission only as far as that set, installed by the same vote, is the Foundation's (§6.4).
10. **Regeneration.** While the chain runs on a single node its operator can regenerate it under the specification's rule, and has done so four times (24 September, 26 September, 4 November and 22 November 2026), each retired chain

kept as test data in an archive that rests on the operator's record and is published with the chain-continuity pages; no credential or record survives a regeneration, a further one is clean and not dated, and its 14-day notice is the operator's commitment, not a rule of the specification, except for a regeneration the chain's security requires at once (§6.4).

11. **The two parts of the dispute key behind one operator,** kept in infrastructure it runs, and today behind one natural person's credentials; at a later clean regeneration, which follows the custodian's key ceremony, the specification moves the Foundation's part to a custodian that operates no issuer, chosen by the MintID Foundation, a related party of the operator, whose key ceremony has two approving parties, the MintID Foundation and its founder, the operator's beneficial owner, so that until the compliance council is seated with several persons one person can still reach both parts; the openings at the verifiers the operator runs stay public and counted, and within the operator's reach (§4.6, §9.3).
12. **The relay and the issuer under one operator,** behind one load balancer; a transport that hides the holder from that operator, and relays run by others, are due before the first node the operator does not run serves holders (§3.8).
13. **Self-revocation dependencies.** Acknowledgement without removal is visible only to the holder, and the holder can record it as the issuer's fault within the evidence window for an issuer that leaves the credential out of its committed removals, not for one that lists it and keeps it; under the public status log an issuer withholds a witness only when it publishes none or the holder's last update is older than the publication it keeps; an agent's cached witness lasts only the ring, about four minutes on the single-node production chain, after which withheld witnesses block it; the validator can delay or censor a submission, and a governed value voted by the single validator can switch self-revocation off; a code is not unique to one credential, so its record may name another credential: the holder library, as released, counts only a record that carries its own message's ciphertext, the remedy otherwise is the issuer's revocation on request, and proving the code's relation to the link secret, which would prevent a record under a credential whose link secret does not give the code, is deferred until an external review of the self-revocation proof; the confirmation read is not proven (§3.8, §8.12).
14. **The emergency key acts alone.** Only a per-block limit bounds its use, and no seated body reviews it (§3.7).
15. **Real-time relay of a genuine challenge.** The holder library checks the challenge's signature, not the channel; stopping a relay to the issuing verifier is the host's. On the x402 transport with Web Bot Auth, a presentation submitted under a key other than the one its challenge was issued to is refused (released, in the x402 middleware and the agent clients; built and switched off in the verifier service in production); relaying the agent's whole signed request stays the channel's (§3.5, §4.3).
16. **Unaudited core, unaudited unlinkability.** The implementation audit and the self-revocation audit are scheduled; the dispute-ciphertext audit has its scope written and is not commissioned; the composition of the presentation, the three new compositions (§3.1), the pseudonym's blinding generator, the derivation of the signature scalar (Table 3), the public status log, the hiding its padding gives and the holder's update from it (§3.7), the BLS24-509 curve code and MintID's BBS ciphersuite on it (§3.1), the status tree's construction (Table 4), the link secret's hiding and every unlinkability claim are audit items, and none has been independently analysed; the project's written soundness and witness-indistinguishability arguments for the self-revocation proof, its written argument for the pseudonym statement with the identity point as blinding generator, and its draft argument for the dispute ciphertext are not yet published and have not been reviewed outside it; a reproducibility run of the record layer's verifier on its native platform is pending (§3.8, §4.6, §11).
17. **A leaking holder prover is undetectable.** Every hiding property rests on the holder's prover drawing fresh randomness, and a verifier's checks are the same for a prover that does not, so a verifier cannot detect a holder implementation that leaks. The published holder builds carry a credential library whose provers are within the scope of the scheduled implementation audit; the claims of this paper cover those builds only (§8.4). The holder

library checks its own prover's randomness before a proof leaves it, the self-revocation prover included; the check refuses known faulty patterns, does not show that randomness is fresh, and protects only the holders that run the library.

18. **Retention.** The dispute retention record's window is a governed parameter of the record layer, read by the verifier with its proof; where the record layer does not carry the window, the verifier's own setting is held within the range allowed to the governed one; the window's length and the record's data-protection framework are open; block history keeps self-revocation ciphertexts until pruned, and the chain's retention policy is open (§4.6, §6.4).
19. **Disclosure-channel residuals.** Correlation is intrinsic to any release. The issuer sees which relying party asked about which of its agents. The issuer is trusted for the person's decision in consented commercial disclosure and mandated identification: the record of the person's choice, the consent device, the release and the receipts are the issuer's. Under the specification the issuer that keeps the record of the person's choice is also the party that may take a brokerage fee on an accepted bid (not built); today that is the first issuer, which the operator runs (§9.3). A delegated release can reach a list by category or membership, and a mandate every counterparty of an obligation class. As built, the class and the rules arrive in the enrolling platform's mint request, and the person is not shown them before the mint (§7.2). The specification prices consented commercial disclosure (§7.4). A relying party sets its own anti-coercion flag (§7.4, §7.5).
20. **Parties outside the protocol.** Whoever runs an agent or enrols it for the person may know its link to the person, and the issuer, its identity-verification provider and whoever holds a part of the dispute key remain subject to the law that applies to them (§2.3, §7.1).
21. **The classical seller rail is not zero-knowledge** (§5.1).
22. **No published holder host keeps the Foundation's key record.** The published holder hosts check each Foundation key record they are given and keep none, so a holder does not notice a second record for the same key version, which would tag the agents given it (T-F); a verifier keeps the first record body it accepts for each key version and refuses another, and accepts the same body signed by another active key of the admission key set; from a later clean regeneration, which follows the custodian's key ceremony, a holder also keeps the first record body it accepts for each version, for the life of its process (Table 3, §4.6).
23. **The curve in production is below the specified security level.** BLS12-381 reaches about 126 bits in its target group, about 127 for generic discrete logarithms and about 110 bits for BBS signatures and the accumulators at 2^{30} signatures under one key, short of the 128 bits the specification requires; the move to BLS24-509 is specified, has not started and carries no date, and until a clean regeneration carries it every deployment runs on BLS12-381. No implementation of BLS24-509 has been audited, and MintID's own would be new code in every holder and in block execution. Nothing in the construction is post-quantum (§3.1).
24. **Sessions and tokens built on an accepted presentation.** They are part of that acceptance, and the person answers for their use. An access token that its resource server validates without introspection lasts until its expiry, at most the maximum accepted age of a status commitment after it was issued, so it can be used after the effect instant of a revocation; the agent-to-agent handshake's re-check, which ends its session when a fresh presentation would be refused on the status commitment its acceptance rested on, is specified and released, in the verifier library, which no deployment runs (§4.3, §6.3).

9.3 The operator and related parties

MintID is developed by Conectia OÜ (Estonia, registry code 16847069), trading as Conectia PRO, one company, which operates the single-node production chain under the mandate of the MintID Foundation. The founder of MintID, Marc Molas, is the beneficial owner of Conectia OÜ and the President of the MintID Foundation; he is recused from every discretionary decision of the Foundation that concerns Conectia; admissions under the published policy leave

nothing to discretion, and he executes them as a ministerial act, the admission of Conectia's own verifier included, each marked "related party" in the public admissions log. On the single-node production chain Conectia runs the one validator, the node verifiers read, the first accepted issuer, the privacy relay and the only admitted verifier, and executes the Foundation's key ceremonies, a related-party position the project discloses; ZadQ, the first product built on MintID, is a commercial service of Conectia. The separation of parties that this paper's properties describe is a property of the design; on production those parties stand behind one company and its founder, and the Foundation is not an independent check on the operator. No claim of independence between these parties is made. The companion states the related-party roles and flows of the project's development, economics and governance.

10

What is not claimed

> What this paper does not say, so that no reader infers it.

10.1 Statements this paper does not make

The limits stated with each property (§8) and the residuals of §9.2 are statements this paper does not make, and they are not repeated here. Beyond them:

1. An agent presentation proves the issuer's attestation at mint, which is what the specification requires of it, and proves nothing about the principal's credential at presentation time; a revocation of that credential reaches the agents through the issuer's cascade, within its bound (§4.8, §6.3).
2. A presentation does not combine several credentials: one policy is answered by one credential.
3. No national eID scheme is named in this paper, and none is stated as a route.
4. The credential does not carry its disclosure class.
5. A relying party cannot require that identity be revealed as a condition of accepting an agent, except, on the mandate basis, for an act a legal obligation attaches to; that basis is built and switched off.
6. No requirement specifies a disclosure to any relying party and nothing of it is built; it would change the specification and stands in tension with the definition's "without revealing who that person is".
7. The protocol has no object for an identification route, so no verifier can admit or refuse one; one would need an issuer identifier per route or a route claim in the credential, and would narrow the anonymity set (§5.5).
8. Nothing in this paper is independently audited.
9. The construction runs on BLS12-381 and, under the specification, on BLS24-509. Nothing in the construction is post-quantum: an adversary able to compute discrete logarithms on these curves would forge credentials and proofs, and could open recorded ciphertexts and link an agent's pseudonyms after the fact, including those kept in block history. No system-wide level of 128 bits is claimed for any deployment before the move to BLS24-509 reaches it (§3.1).

10.2 Directions

A direction is stated only with its status and never in the present tense of the protocol.

- **Disclosure under conditions set at issuance.** Within the consent rules and the mandates, built and switched off; the class committed in the credential, specified; conditions that would release identity without the person's own decision, outside a legal obligation, excluded by the specification today (§7.7).
- **KYB.** We are building KYB.
- **A proof of the principal's standing in the agent presentation.** Parked until the design of agent profile v3: a version 3 of the agent profile that carries, hidden, the status reference of its principal's credential, and an agent presentation that proves that reference unrevoked under the issuer's current status commitment. It would prove one of the properties the issuer attests today, and a principal's revocation would reach the agents at the bound of the principal credential's own path, independently of the cascade. It would also place one hidden value in every agent of a person and make a person's agents fail at the same commitment, so it is taken up only with a demonstration that a person's agents stay unlinkable to each other (§8.10).

10.3 Outside the protocol

The dispute rail has no financial layer: no party of the protocol holds, locks or reserves funds for a claim, and a settlement is paid by the person to the claimant directly, by the person's own means. The payment for which an agent proves its identity is outside MintID. No party of the protocol settles, holds or moves it, and no payment instrument or payer is bound to a credential or to a pseudonym. The one link the specification allows is a digest of a payment's terms that a verifier may keep beside a presentation (§4.3; specified; released, in the x402 middleware and the agent clients; built and switched off in the verifier service in production). What the payment rail itself reveals, such as a payer's address on a public ledger or the card details a seller receives, stays a property of that rail. The pricing of consented commercial disclosure is stated in §7.4. MintID proves agent identity and claims neither provenance, runtime evidence nor workload identity: a manifest, a run record or a workload identifier proves what ran and names a process, never who answers for the agent. An organisation as the principal behind an agent is outside the protocol today. Delegation from one agent to another is deferred. Offline verification is not supported (§3.6).

11

Status

> One table, in the roadmap's words.

Every statement about a piece that is not in production carries its status where it is made, in the status words that §1.3 defines. The present tense without a status word is reserved for what is in production. A direction appears only with its status and never in the present tense. Table 13 gathers every piece; where it and a sentence of the paper differ, the table governs, and the table follows the project's roadmap on the day of writing. Its last column names the section of the roadmap, a document of the specification corpus available on request, not a section of this paper.

Table 13 · Status.

PIECE	STATUS	ROADMAP SECTION
Identity credential: blind issuance, presentation with predicates and non-revocation	In production	Roadmap §1.2, proof core
Agent credential (profile v1): mint from a principal presentation, agent presentation with scope, custody and limit predicates	In production	Roadmap §1.2, proof core
The disclosed claims of every presentation bound to the values its signature proof proves, so that the status reference a mint or relay presentation discloses to the issuer is that of the credential it proves	In production; specified for the reference a mint presentation discloses	Roadmap §1.2, proof core and issuer service
Agent credential profile v2 (the principal's age floor)	Built, not issued: the issuer has not switched its mint to it; the single-node production chain carries its self-revocation entry	Roadmap §1.2, proof core
Pairwise pseudonym, mandatory in agent policies	In production	Roadmap §1.2, verifier service
The unexpired predicate asked by every policy, a policy without it refused	In production	Roadmap §1.2, verifier service; §1.4
Binding of a presentation to one payment by a digest of the payment's terms, kept by the verifier (the carriers its profile names: x402 version 2; the Machine Payments Protocol; the Agentic Commerce Protocol; the Universal Commerce Protocol; the Agent Payments Protocol with a Universal Commerce Protocol checkout)	Specified; released, in the x402 middleware and the agent clients; built and switched off in the verifier service in production; its profile, which names the carriers, accepted and in force; that a dispute claim names the payment its presentation was bound to, specified; released, in the verifier library; that no payment instrument or payer is bound to a credential or a pseudonym, and that no party of the protocol settles a payment, specified	Roadmap §3.6, x402 integration decision register, row D-x8
Dispute ciphertext in every agent presentation; the dispute retention record	In production	Roadmap §1.2, §1.4
The signature scalar of blind issuance derived as the blind BBS draft derives it; the refusal of a status reference the issuer's key set has already signed	Decided; in production; the derivation pending audit (the implementation audit, scheduled)	Roadmap §1.4, issuance and the holder library
The holder library's check of its own prover randomness before a proof is released	Decided; released, in the holder libraries	Roadmap §1.4, issuance and the holder library

PIECE	STATUS	ROADMAP SECTION
Release of the MintID Foundation's share after a recorded claim	Specified; in the tree of the release, run by no deployment	Roadmap §1.4
The issuer's completion	Specified; in production's tree and called by no route	Roadmap §1.4
The claim recorded on the record layer, and each release of the Foundation's share published there before it is made, with the Foundation's part in infrastructure the operator runs	Specified; in production on the record layer; the verifier's filing of a claim released, in the verifier library	Roadmap §1.4, the dispute rail
The claim window as a governed parameter of the record layer, read by the verifier with its proof	Specified; in production, 90 days at genesis; its length open; on a chain that does not carry the window, the verifier's own setting held within the range allowed to the governed window	Roadmap §1.4, the dispute rail; §1.7
The MintID Foundation's part of the dispute key held by a custodian that operates no issuer, chosen by the MintID Foundation, a related party of the operator, with its co-signed key record	Specified, for a later clean regeneration, which follows the custodian's key ceremony; the custodian and its contract open; the ceremony's two approving parties the MintID Foundation and its founder	Roadmap §1.4, the dispute rail; §1.7, custody separation
A verifier's pin on the body of the MintID Foundation's key record, the same body accepted under another active key of the admission key set	Specified; in production	Roadmap §1.4, the dispute rail
Claim with its proof of harm, notice, private settlement, arbiter registry and sortition, the arbiter's offer, fraud signal, identity release	Specified	Roadmap §1.4
Status commitments every 30 s, emergency commitments, ring of recent commitments, acknowledgement rule	In production	Roadmap §1.2, chain modules
Holder self-revocation: the proof verified by the chain, the privacy relay, the issuer's fold, the cascade spread, cover traffic	In production	Roadmap §1.2, self-revocation
A public, padded log of each accumulator's removals, from which a holder updates its status witness without naming its credential	Specified; in production, the holder's update from it released in the holder libraries; the hiding its padding gives pending audit (the privacy review of the public status log, not started)	Roadmap §1.2, issuer service; §1.4, the public status delta log; §2.6
A holder's evidence of an acknowledgement without removal, and the issuer's fault recorded on the record layer	Specified; the record layer's part in production, the holder's part released, in the holder libraries; until the council is seated, a suspension on an uncorrected fault decided by the MintID Foundation under its admission policy, without discretion, and executed by the operator, whether a related party may execute it for the operator's own issuer a counsel item and open; the correction period 24 hours at the reference values; the freeze on a recorded fault pending audit (the self-revocation audit, scheduled)	Roadmap §1.4, non-removal evidence and issuer fault; §1.7
Holder transport hiding the holder from the operator; relays run by others	Scheduled: due before the first node the operator does not run serves holders	Roadmap §2.5
Revocation bounds: one block, 185 s, 245 s, 395 s on the single-node production chain	In production	Roadmap §1.2
Verifier service: acceptance conditions, single-use challenge, proven reads	In production	Roadmap §1.2, verifier service

PIECE	STATUS	ROADMAP SECTION
Sessions and access tokens built on an accepted presentation, as part of that acceptance: the x402 session's re-check on every reuse, an access token's lifetime bounded by the maximum accepted age of a status commitment	Released, in the software a verifying party runs, on the verifier service's status-commitment check, which is in production; the agent-to-agent handshake's re-check specified; released, in the verifier library, which no deployment runs	Roadmap §1.2, verifier service, agent channel and the GTM wave of 2026-10-29; §1.4
The session binding on the x402 and Web Bot Auth transport: a challenge issued to the key that signed its request, its presentation accepted only under that key, an unsigned agent refused by default	Released, in the x402 middleware and the agent clients; its default mode decided; built and switched off in the verifier service in production	Roadmap §1.2, agent channel; §1.4, the session binding
Which status commitments a verifier accepts: for an agent presentation, the issuer's latest finalised commitment or the one it superseded; for an identity presentation, the latest, whatever commitment the presentation names; for a session or an access token, the rule of its profile; the holders' fetch schedule, common to the holders of a profile	Specified; in production in the verifier service, released in the verifier and holder libraries; the identity holders' default fetch period, 300 seconds, decided and released	Roadmap §1.4, which status roots a verifier accepts
A thirty-second presentation lifetime, one constant of the presentation format, released to every holder and verifier together	Specified; in production; live	Roadmap §1.4, the 30-second presentation window
One encoding per presentation, every other refused by every verifier before it reads a value; the dispute presentation digest computed by verification over the bytes it accepted	Decided; in production; live; the completeness of its rules pending audit (the implementation audit, scheduled)	Roadmap §1.4, the 30-second presentation window
At least 128 bits of classical security; every pairing-based statement on BLS24-509	Specified; not started, no date; built before the audits and carried by a clean regeneration	Roadmap §1.4, the 128-bit security level and BLS24-509
Verifier admission: the Foundation's grant required by the chain's rules on both tiers, at a verifier's self-registration and self-renewal, at the admitted tier's registration, on every other message of the chain's authority that changes a verifier's record and on every verifier record of a genesis, each checked against the admission key set, which the chain's governance installs (for a genesis record, the set that genesis carries), so that a grant shows the Foundation's admission only as far as that set is the Foundation's; the admitted tier registered through the governance path voted by the single validator	In production	Roadmap §1.2, chain modules, operations and verifier admission; §1.4, the Foundation's grant on the admitted tier and on the authority's other messages
Issuer admission: decided by the MintID Foundation, executed as a governance proposal voted by the single validator	In production, not yet used on the current chain: its one issuer, the operator's, was written at its genesis	Roadmap §1.2, operations
A dated history of each issuer's keys on the record layer, read with a proof	Specified; in production	Roadmap §1.2, chain modules; §1.4, issuer key histories
A third party's own node synced from the single-node production chain: the peer entry, open once applied to the nodes of admitted integrators only	Built, not applied: built in the deployment code and live on the public testnet; not answering on production	Roadmap §1.3, the production peer entry
Compliance council	Built into the chain, not seated; its seating specified, not built	Roadmap §1.4
Classical seller rail (issuer-signed attestations, never zero-knowledge), used by ZadQ's production clients	In production	Roadmap §1.2

PIECE	STATUS	ROADMAP SECTION
Identification through the operator's identity-verification provider (grounds A2; A3 not issued on production)	In production	Roadmap §1.2, issuer service and GTM wave
The AML labels of A3 and A4 in the issuer policy	In production; live	Roadmap §1.2, issuer service since 2026-11-10
The European Digital Identity Wallet's person identification data as an evidence route at the issuer	Built and switched off in every deployment (the issuer's driver); the rest decided, not specified	Roadmap §1.3, §1.4
KYB	We are building KYB; not in production	Roadmap §2.1
Consented commercial disclosure, with the offer's bid reported as payable	Built and switched off; the delivery channel specified, the issuer's delivery and its directory of relying parties built and switched off with the rail, and the receiving side built and switched off; no release can yet complete in a deployment	Roadmap §1.2, issuer service; §1.4; §2.1; the bid from the code
The items and the delivery of consented commercial disclosure: a contact attribute, then the person's identity without the identity document's number and type, and no pseudonym item; delivery to the origin at which the request's presentation was made; a release whose body was written and that the relying party refuses, answers outside the delivery contract or does not answer in time receipted as delivered and refused, with no bid payable	Specified; built and switched off with the rail	Roadmap §1.4
Settlement of a disclosure bid, its protocol cut, a brokerage fee	Specified, not built	Roadmap §1.4
Mandated identification	Built and switched off, behind its own switch; the governed schedule on chain specified and in production, its three classes drafts at genesis, none in force, a class once withdrawn staying withdrawn and a new obligation taking a new class; the issuer serving only the classes the schedule governs built and switched off with the rail, so the issuer in production serves none; the delivery channel specified, the issuer's delivery and its directory of relying parties built and switched off with the rail, and the receiving side built and switched off; no release can yet complete in a deployment	Roadmap §1.2, §1.4
Disclosure-class claim in the credential	Specified, not built; the credential profile that carries it open	Roadmap §1.4, §1.7
Principal control surface	Built and switched off in every deployment	Roadmap §1.2, issuer service
Disclosure to any relying party	Not specified: no requirement specifies it and nothing of it is built	none
A per-person, per-verifier tag across a person's agents	Open	Roadmap §1.7
A proof in the agent presentation that the principal's credential is unrevoked	Parked until the design of agent profile v3	Roadmap §1.7, a proof by the agent that its principal credential is unrevoked
Delegation from one agent to another	Deferred	Roadmap §2.6

PIECE	STATUS	ROADMAP SECTION
Running the protocol on several nodes	Not built; foreseen as the project evolves	Roadmap §1.5, §2.1
The public testnet with its KYC sandbox, the only deployment where a third party may test	Live, online during announced test rounds and on request	Roadmap §1.2
Implementation audit of the proof core	Scheduled	Roadmap §2.1, §2.6
Holder self-revocation audit	Scheduled	Roadmap §2.1, §2.6
Dispute-ciphertext audit	Scope written	Roadmap §2.6
Privacy review of the public status log	Not started	Roadmap §2.6
Proof-of-harm circuit audit	Not started	Roadmap §2.6

GLOSSARY

Glossary

Accepted issuer: a party admitted under the MintID Foundation's published policy that identifies persons and issues credentials. **Accountability escrow**: the issuer's private record linking an agent to its person; it holds a link, never funds. **Arbiter**: the party that decides a claim and certifies the finding on which the issuer releases identity (specified). **Authenticated read**: a value read from the record layer with a proof against the state commitment of a node the reader designates; a verifier designates a node it runs or a node of the operator; the light-client verification is not built (§6.4). **Classical seller rail**: issuer-signed attestations read by a standing read, never zero-knowledge. **Claim window**: how long a verifier keeps the dispute retention record; a governed parameter of the record layer (§4.6). **Custodian**: under the specification, from a later clean regeneration, which follows the custodian's key ceremony, a party that operates no issuer, chosen by the MintID Foundation, a related party of the operator, that holds the Foundation's part of the dispute key and co-signs its key record (§4.6, §9.3). **Decision record**: the verifier's own record of a decision, closed, minimal and never published. **Disclosure channel**: the paths along which identity detail can leave the issuer, the only party that, within the protocol, can join an agent to its person, once a credential exists (§7). **Dispute ciphertext**: every agent presentation to a registered verifier carries an encryption of the agent's status reference under the dispute key, fresh per presentation and unreadable by the verifier; neither the issuer nor the Foundation can open it alone, and the Foundation adds its share, as a ministerial act, only for a claim recorded on chain; the ciphertext in production, the claim record on chain specified and in production, the release of the Foundation's share after a recorded claim specified, in the tree of the release and run by no deployment, and the issuer's completion specified, in production's tree and called by no route (§4.6, §7.3). **Dispute key**: held jointly by the issuer and the MintID Foundation; in production, both parts kept today in infrastructure the operator runs (custody: §4.6). **Dispute retention record**: the verifier's own record of an accepted agent presentation, kept for the claim window and then deleted, never published. **Effect instant**: the bound of a revocation's path (§6.3). From it every conforming verifier refuses the revoked credential while the issuer removes what the path requires it to remove and, on the cascade, builds a commitment every heartbeat; on the holder path, an acceptance after it that rests on a commitment that acknowledged the code without removing the credential is, under the specification, the issuer's fault, not the holder's, and on the principal-wide cascade an acceptance after it that rests on a commitment without the agent's removal is, under the specification, the issuer's fault, not the principal's, whether the issuer removed the agent late or did not build a commitment every heartbeat; no record shows that a commitment kept such an agent (§8.7). A session or an access token built on an acceptance before it is part of that acceptance (§4.3). **Enrolling platform**: the party that requests an agent's mint for the person, an operator of the commercial layer; not the operator of the chain. **Evidence route**: how an issuer identifies a person, described and graded by its published policy. **Freshness band**: F1 to F4, published by the issuer and read on the classical seller rail; not a presentation predicate. **Holder library**: MintID's headless holder software, with Python, Go and TypeScript bindings. **Holder self-revocation**: the holder's own revocation by a one-time code and a proof the record layer verifies; in plain language, the holder's kill switch: it revokes the credential's authority; it does not halt the agent runtime (§3.8). **Link secret**: the holder's secret, signed blind and never revealed. **Presentation rail**: the zero-knowledge presentation, in production, whose audit is a scheduled milestone. **Principal**: the person behind an agent. **Principal control surface**: the issuer's passkey-authenticated page for the person it verified; the holder library stays the sovereign path (built and switched off; §4.8). **Privacy relay**: the service that carries and pays for self-revocations and, at its application layer, never holds an address with a code. **Record layer**: the role the chain plays for the protocol, the shared record from which issuers, holders and verifiers read the public facts they rely on (the registries of accepted issuers and verifiers, the issuers' status commitments, the one-time codes of holder self-revocation) and which applies the rules under which those facts are written; no presentation is ever written to it, and a verifier only reads it to decide (§6). **Relying party**: a verifier, as it appears in the disclosure channel. **Single-node production chain**: the MintID chain in production,

operated by Conectia under the MintID Foundation's mandate. **Standing read**: the classical seller rail's read of an attestation's state. **Status commitment**: an issuer's signed 30-second commitment covering its accumulators; a credential proves itself unrevoked by membership, for which the issuer's witness serves any credential it has not removed (§3.7). **Status reference**: the hidden claim by which a credential is a member of its issuer's accumulator. **Status witness**: the holder's material for proving membership, updated by the holder from the issuer's public status log, or refreshed from the issuer when the log cannot serve it (§3.7). **The chain**: MintID's record layer today, a single-node chain operated by Conectia under the MintID Foundation's mandate. **The MintID Foundation**: holds the protocol's authority, admitting issuers and verifiers under a published policy; it issues nothing, certifies nothing and verifies nobody. **Verifier**: the role that decides on its own service, admitted under the published policy with a responsible person identified.

REFERENCES

References

The list gives the documents the text cites. Internet-Drafts are work in progress, cited at the revision current on the date of this paper; software is cited by its maintainers and public location, without a version.

Signatures and credentials. D. Boneh, X. Boyen, H. Shacham, “Short Group Signatures”, CRYPTO 2004 (the BBS proposal) · M. H. Au, W. Susilo, Y. Mu, “Constant-Size Dynamic k-TAA”, SCN 2006 (BBS+, the variant with a further signature scalar) · J. Camenisch, M. Drijvers, A. Lehmann, “Anonymous Attestation Using the Strong Diffie Hellman Assumption Revisited”, TRUST 2016 (BBS+) · S. Tessaro, C. Zhu, “Revisiting BBS Signatures”, EUROCRYPT 2023, IACR ePrint 2023/275 · T. Looker, V. Kalos, A. Whitehead, M. Lodder, G. M. Bernstein, “The BBS Signature Scheme”, draft-irtf-cfrg-bbs-signatures-14, 29 October 2026 · V. Kalos, G. M. Bernstein, “BBS per Verifier Linkability”, draft-irtf-cfrg-bbs-per-verifier-linkability-03, 29 June 2026 · V. Kalos, G. M. Bernstein, “Blind BBS Signatures”, draft-irtf-cfrg-bbs-blind-signatures-05 · AnonCreds v2 and its credx implementation, Hyperledger AnonCreds developers, github.com/hyperledger/anoncreds-v2-rs · J. Camenisch, A. Lysyanskaya, “An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation”, EUROCRYPT 2001 (pseudonym systems) · S. Brands, *Rethinking Public Key Infrastructures and Digital Certificates*, MIT Press 2000 (U-Prove) · D. Boneh, X. Boyen, “Short Signatures Without Random Oracles”, EUROCRYPT 2004 (q-SDH).

Accumulators, commitments, range proofs. G. Vitto, A. Biryukov, “Dynamic Universal Accumulator with Batch Update over Bilinear Groups”, IACR ePrint 2020/777 · T. P. Pedersen, “Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing”, CRYPTO 1991 · B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, G. Maxwell, “Bulletproofs: Short Proofs for Confidential Transactions and More”, IEEE S&P 2018, IACR ePrint 2017/1066; the implementation over BLS12-381 used, github.com/zkcrypto/bulletproofs.

Proofs, transcripts, hashing, curves. A. Fiat, A. Shamir, “How to Prove Yourself: Practical Solutions to Identification and Signature Problems”, CRYPTO 1986 · C. P. Schnorr, “Efficient Signature Generation by Smart Cards”, Journal of Cryptology 1991 · D. Chaum, T. P. Pedersen, “Wallet Databases with Observers”, CRYPTO 1992 · D. Pointcheval, J. Stern, “Security Arguments for Digital Signatures and Blind Signatures”, Journal of Cryptology 2000 · M. Hamburg, “The STROBE protocol framework”, IACR ePrint 2017/003 · Merlin transcripts, the dalek-cryptography project, github.com/dalek-cryptography/merlin · gnark-crypto, Consensys, github.com/consensys/gnark-crypto · RFC 9380, Hashing to Elliptic Curves · FIPS 180-4, Secure Hash Standard (SHA-256) · FIPS 202, SHA-3 Standard (SHAKE256, Keccak) · RFC 2104, HMAC: Keyed-Hashing for Message Authentication · Y. Sakemi (Ed.), S. Kanno, R. S. Wahby, “Pairing-Friendly Curves”, draft-irtf-cfrg-pairing-friendly-curves-15, 1 November 2026 (BLS12-381) · D. F. Aranha, G. Fotiadis, A. Guillevic, “A short-list of pairing-friendly curves resistant to the Special TNFS algorithm at the 192-bit security level”, IACR Communications in Cryptology 1(3), 2024, IACR ePrint 2024/1223 (BLS24-509) · European Commission, “Specification for the implementation of Zero-Knowledge Proofs based on multi-message signatures in the EUDI Wallet” (TS14), version 1.0, 27 February 2026.

Encryption. T. ElGamal, “A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms”, IEEE Transactions on Information Theory 1985 · C. P. Schnorr, M. Jakobsson, “Security of Signed ElGamal Encryption”, ASIACRYPT 2000. The proof of correct encryption is a Schnorr-style proof over ElGamal in G_1; no separate reference is given for it.

Signatures, trees and state in the record layer. RFC 8032, Edwards-Curve Digital Signature Algorithm (EdDSA) · RFC 6962, Certificate Transparency, obsoleted by RFC 9162, Certificate Transparency Version 2.0 · ICS-23 store proofs, github.com/cosmos/ics23 · Cosmos SDK, github.com/cosmos/cosmos-sdk · CometBFT, github.com/cometbft/cometbft.

Transports and related work. OpenID for Verifiable Credential Issuance 1.0, Final, 16 September 2025 · OpenID for Verifiable Presentations 1.0, Final, 9 July 2025 · OpenID Connect for Identity Assurance 1.0, Final · RFC 8693, OAuth 2.0 Token Exchange · RFC 9458, Oblivious HTTP · RFC 9421, HTTP Message Signatures · Web Bot Auth, IETF working-group Internet-Draft (HTTP Message Signatures for automated traffic), work in progress · x402, the payment protocol of that name, maintained by the x402 Foundation · Machine Payments Protocol (MPP), the Payment HTTP Authentication Scheme, IETF Internet-Draft, work in progress · Agentic Commerce Protocol (ACP), specification of 17 April 2026 · Universal Commerce Protocol (UCP), release of 25 August 2026 · Agent Payments Protocol (AP2), version 0.2, 28 April 2026 · RFC 9901, Selective Disclosure for JSON Web Tokens (SD-JWT), §10.1 · SD-JWT-based Verifiable Credentials (SD-JWT VC), IETF OAuth working group Internet-Draft, work in progress · ISO/IEC 18013-5, mobile driving licence (mdoc).

Assurance and law. Regulation (EU) No 910/2014 as amended by Regulation (EU) 2024/1183 (eIDAS), Art. 8 · ISO/IEC 29115, Entity authentication assurance framework · ETSI TS 119 461 V2.1.1, identity proofing of trust service subjects · NIST SP 800-63-4, Digital Identity Guidelines · Regulation (EU) 2024/1624 (AMLR), Arts. 20, 22(6)(b) · Regulation (EU) 2023/1114 (MiCA).

MintID. The specification corpus (the core specification of the protocol, the agent-identity specification, the disclosure specification, the decision records and the project’s roadmap), available on request · the formal specifications, published with the source under the Apache-2.0 licence · the companion, *Governance, Economics and Launch Plan* v1.0 · the Issuer, Verifier & SDK, Holder and Validator manuals.

APPENDIX A

Requirements and where this paper treats them

The core specification states 28 requirements. Table A1 lists them with the section of this paper that treats each, or the companion where the requirement belongs to the chain's design, its economics or its governance. The wording of the requirements is the specification's, and where it names KYC it names the identification an accepted issuer performs under its published policy; this paper's own text uses the identification vocabulary of §5.

Table A1 · The core requirements.

ID	REQUIREMENT	TREATED IN	ID	REQUIREMENT	TREATED IN
R1	The chain's consensus and finality (design)	the companion	R15	Online verification against the latest finalised state	§3.6, §6.2
R2	No raw KYC on-chain	§2.3, §8.14	R16	Issuer bonds (design)	§3.8 (the freeze); the companion
R3	User custody of credentials and secrets	§2.1	R17	Monetary policy (design)	the companion
R4	Multiple credentials, no public master identifier	§8.10	R18	Privacy-preserving verifier workflow; presentations never on-chain	§2.2, §3.6
R5	Accepted issuer model (admitted under the published policy)	§5, §6.4	R19	Bounded state growth: issuers, verifiers, validators; never holders	§2.3, §8.14
R6	Private assurance grades A1–A4, threshold proofs	§3.4, §5.3	R20	Security-first release: audits, adversarial testnet, bounty, incident plan	the companion, §11
R7	One-time issuance; renewal creates a new credential	§3.9	R21	Selective disclosure and predicates *(should)*	§3.4
R8	Root-based status, no per-user state	§3.7	R22	Multi-claim sessions *(should)*	§3.4, §10.1 (one policy, one credential)
R9	30-second status freshness, fail closed	§3.7, §6.3	R23	Holder-side trusted verifier domains *(should)*	§3.5
R10	Immediate issuer emergency revocation via an emergency key	§3.7	R24	Verifier domain-control revalidation, no wildcards *(should)*	§6.4
R11	Holder self-revocation revealing neither identity nor credential identifier	§3.8, §8.12	R25	Independent evidence commitments for council decisions *(should)*	the companion
R12	Recovery only by KYC re-verification	§3.9	R26	Optional paid services, firewalled from consensus rewards *(should)*	the companion
R13	KYC-approved verifiers bound to domains and keys	§6.4	R27	State pruning and snapshots *(should)*	the companion
R14	Thirty-second, fully bound, perishable presentation	§3.5, §8.8	R28	First-party KYC integration behind a provider-neutral interface *(should)*	§5.2

The agent-identity and disclosure specifications state further requirements, of which this paper treats the following (Table A2); each line names the requirement, and titles that belong to the design are glossed.

Table A2 · The agent-identity and disclosure requirements cited.

ID	REQUIREMENT	TREATED IN	ID	REQUIREMENT	TREATED IN
R-A1	Backing is attested and proven; identity is not revealed	§4.2 (issuer-attested at mint), §4.8	R-A16	Rail segregation	§7.1
R-A2	Sibling unlinkability	§8.10	R-A17	Identity-last dispute cascade	§7.3
R-A3	The only link lives in escrow (the accountability escrow, the issuer's private record)	§4.2, §7.1	R-A18	No consent on the dispute rail	§7.3
R-A4	Issuer-mediated minting, no new chain message	§4.2, §4.9	R-A19	Disclosure class committed at mint	§7.2, §7.7
R-A5	Delegated-scope predicates	§4.4	R-A20	Graduated, field-minimal disclosure	§7.4
R-A5.1	Five-axis scope vocabulary (v2)	§4.4	R-A21	Consent resolution (commercial rail)	§7.4
R-A5.2	Key-custody-class attribute	§4.3	R-A21.1	Delegated-with-limits consent (tiered ceiling)	§7.4
R-A6	Per-agent kill switch	§4.8	R-A22	Bid vs. fee	§2.3, §7.4; the companion
R-A7	Sybil resistance (the issuer's cap; fees in the design)	§4.2; the companion (the fees)	R-A23	Anti-coercion	§7.4, §7.5
R-A8	Accountability escrow and disclosure governance	§2.3, §7.1	R-A24	Correlation transparency and minimisation	§7.4
R-A9	Bounded state and consensus determinism preserved	§2.3, §4.9, §8.14	R-A25	Disclosure receipts	§6.2, §7.4
R-A10	Pairwise agent pseudonym	§4.5	R-A26	Reserved	none
R-A10.1	Dispute ciphertext	§4.6	R-A27	Dispute adjudication (notice, private settlement, arbiter's offer, zero-knowledge proof-of-harm floor)	§7.3
R-A11	Agent-to-agent verification mode (the handshake, provisional)	§4.3	R-A28	Arbiter market	§7.3
R-A11.1	Two-tier verifier model and the claim on the dispute rail	§6.4, §7.3	R-A29	Per-agent identification, priced per disclosure under the specification (§7.4); no continuity	§7.4
R-A12	Standards bridging (the token exchange and the transport profiles; no payer bound, §10.3)	§4.3, §10.3	R-A30	Mandated identification	§7.5
R-A15	Identity-last remediation cascade	§7.3			

APPENDIX B

Time values of the protocol

Table B gives every time value the protocol's bounds depend on. "Governed" means a value of the record layer changed only by a recorded decision that states the bounds it changes; "constant" means a value the specification or the presentation format fixes, outside the record layer.

Table B · Time values.

VALUE	SYMBOL	REFERENCE VALUE	SINGLE-NODE PRODUCTION CHAIN	NATURE
Presentation lifetime	none	30 s	30 s	Constant of the presentation format
Heartbeat of status commitments	T_H	30 s	30 s	Governed
Maximum accepted age of a status commitment	T_A	45 s	180 s	Governed
Maximum future skew of a status commitment	T_K	5 s	5 s	Governed
Grace for acknowledging a one-time code	T_G	60 s	60 s	Governed
Ring of recent commitments a verifier may accept	none	8	8	Governed
Spread of a principal-wide cascade	none	the next 8 commitments	the next 8 commitments	Constant of the specification; an issuer setting in production
Emergency path	none	one block	one block	Follows from finality
Issuer status revocation, $T_A + T_K$	none	50 s	185 s	Follows from T_A, T_K
Holder self-revocation, $T_G + T_A + T_K$	none	110 s	245 s	Follows from T_G, T_A, T_K
Principal-wide cascade, $7 \cdot T_H + T_A + T_K$ after the trigger commitment	none	260 s	395 s	Follows from T_H, T_A, T_K

MintID Protocol Whitepaper v2.0, 23 November 2026 (supersedes v1.3 of 15 November 2026; the chapters of v1.3 that are not protocol are carried in the companion, Governance, Economics and Launch Plan v1.0) · prepared by MintID, developed by Conectia OÜ (trading as Conectia PRO), which operates the single-node production chain under the mandate of the MintID Foundation, the protocol's authority (related parties: §9.3) · non-normative; the specification corpus governs · not a MiCA crypto-asset white paper and not an offer, solicitation or marketing of a crypto-asset · not legal, tax or investment advice.