Zero-Knowledge Age Verification Protocol Using Pedersen Commitments

I am still updating the protocol, maybe some aspects aren't to much clear. I will make it much clearer within tomorrow.

1 Overview

This document describes a **basic** two-step protocol that enables a user to prove in zero knowledge that their age is greater than 18, based on a government-issued digital identity. The protocol uses *Pedersen commitments* and *range proofs* to achieve privacy-preserving age verification.

The steps are:

Step 1: The user obtains a signed Pedersen commitment to their birthdate from a trusted authority.

Step 2: The user later proves to a verifier that the committed birthdate corresponds to an age ≥ 18 without revealing the birthdate or any other information.

2 Preliminaries

Let G be a cyclic group of prime order q with public generators $g, h \in G$ such that $\log_g h$ is unknown. The Pedersen commitment to a value $m \in \mathbb{Z}_q$ with randomness $r \in \mathbb{Z}_q$ is defined as:

$$C = g^m h^r.$$

This property is the key to our age-verification protocol.
A second Preliminary is range proofs. (Bullet proofs)

3 Step 1: Credential Issuance by the Authority

3.1 Goal

Allow a user to obtain a cryptographically verifiable, privacy-preserving digital ID that commits to their birthdate.

3.2 Protocol Description

1) The user computes a Pedersen commitment to their encoded birthdate:

$$C = g^b h^r,$$

where:

- b = user's birthdate encoded as an integer (e.g., days since epoch),
- r = random blinding factor.

2) The user sends C to the authority.

3) The authority send a challenge to the user.

4) The user initialize a proof and send it to the authority, together with identifying information (in a secure and authenticated channel).

5) The authority verifies the proof first, then verifies the user's real-world ID and signs the commitment:

$$\sigma = \text{Sign}_{sk_A}(C, M),$$

where M is metadata (e.g., credential ID, expiry date, revocation handle).

6) The authority returns the credential:

$$\mathsf{Cred} = (C, \sigma, M).$$

Protocol Flow (Interactive)

Notation

- G: cyclic group of order q.
- q: large prime.
- Public generators $g, h \in G$.
- User secret attributes:
  - Birthdate $b \in \mathbb{Z}_q$ (encoded as an integer, e.g., 19980630).
  - Blinding random $r \in \mathbb{Z}_q$ (chosen uniformly by the user).
- Commitment:

$$C = g^b \, h^r$$

- Authority has long-term signing key sk_A, and corresponding public key pk_A.
- $H(\cdot)$: cryptographic hash-to-scalar function (and hash-to-group where needed).
- Metadata M: includes parameters such as

$$\{\text{issuer ID}, \text{issuance date}, \text{expiry}, \text{credential type (age-check)}, \text{revocation handle}\}.$$

- A nonce N is included to prevent replay attacks.

1. User locally picks randomness and computes commitment. The user chooses random $r \in \mathbb{Z}_q$ and computes:

$$C \leftarrow g^b h^r.$$

The user keeps (b, r) secret.

2. User proves to the Authority the binding to an authenticated ID. The user presents their real ID (in-person or via a secure channel) so that the authority can verify that the birthdate value b corresponds to the authenticated identity.

Simultaneously, to bind the same b to C without revealing r, the user runs a zero-knowledge proof of knowledge showing they know (b, r) corresponding to C. This uses the Σ-protocol form:

1. **Prover (User)** picks random $s, t \in \mathbb{Z}_q$, computes

$$A = g^s h^t,$$

   and sends A to the Authority.

2. **Authority** sends a random challenge c (or it is derived via Fiat–Shamir).

3. **Prover** responds with:

$$z_b = s + cb, \quad z_r = t + cr.$$

4. **Authority** verifies:

$$g^{z_b} h^{z_r} \stackrel{?}{=} A \cdot C^c.$$

This verification confirms that the user who presented the ID also knows the opening (b, r) of commitment C.

The authority therefore learns the true birthdate (from the verified ID), while the zero-knowledge proof ensures that the commitment C is bound to the same b without revealing r.

3. Authority Checks Eligibility The Authority verifies the user's physical or digital ID and confirms the claimed birthdate b.

If the verification passes, the Authority prepares to sign the user's commitment.

4. Authority Signs the Commitment and Metadata The Authority constructs a message to sign:

$$S = H(\text{"AgeCredential"} \parallel C \parallel M \parallel N)$$

where:

- M includes metadata such as issuance time, expiry, and revocation handle.
- N is a fresh nonce to prevent replay.

The Authority computes a signature:

$$\sigma = \text{Sign}_{sk_A}(S).$$

Finally, the Authority returns (σ, M) to the user.

5. User Stores the Credential The user stores the credential tuple:

$$\text{Cred} = (C,\ \sigma,\ M)$$

and continues to keep (b, r) secret locally. The Authority never learns the blinding factor r.

What the Credential Achieves

The credential is an Authority-backed binding of a hidden birthdate b to the Pedersen commitment C.

- The authority attests that C commits to the correct birth date, but cannot later read b.
- The Authority's signature σ proves that it verified the user's identity and attested to the birthdate claim at issuance time.
- The user can later prove statements about the committed value (e.g., that age > 18) without revealing b or r.
- The metadata M allows verifiers to check issuance and expiry, and supports revocation if needed.

4 Step 2 – Zero-Knowledge Age Verification Protocol

4.1 Protocol Description

- From Step 1, the user holds:
  $$C = g^b h^r$$
  which is a Pedersen commitment to the birthdate b, along with the Authority's signature σ over C and metadata M.
- The verifier checks the Authority's signature (thereby trusting that the Authority attested that an authentic ID was used to produce C).
- To prove that the user's age > 18 at verification time T (today), compute a threshold:
  $$t = \text{encode}\big(\text{latest birthdate such that age} \geq 18 \text{ as of } T\big).$$
  Concretely:
  $$t = \text{encode\_date}(\text{date} = T - 18 \text{ years}).$$
- The goal is to show in zero knowledge that the committed birthdate $b \leq t$. Equivalently, define
  $$v := t - b \geq 0.$$
- The prover constructs a Pedersen commitment D to v from public data and C, then gives a range proof that $v \in [0, 2^k)$. If v is non-negative and small enough, that proves $b \leq t$.
- All is made non-interactive using Fiat–Shamir or a standard non-interactive range-proof system (e.g., Bulletproofs).

  The verifier:

  - checks the Authority signature σ;
  - verifies the range proof on D;
  - checks optional revocation and expiry metadata M.

  No raw b or r is ever revealed.

Precise Protocol (Non-Interactive)

Notation / Parameters

- G: group of prime order q with public generators g, h; the discrete logarithm problem is hard in G.
- encode_date(d): converts a calendar date to an integer scalar (e.g., days since 1970-01-01), such that values fit well below q.
- Security parameter k: chosen such that 2^k is an upper bound for $(t - b)$. Example: $k = 32$ covers many years.
- **Range proof system:** a non-interactive proof that a committed value lies in $[0, 2^k)$. We call these algorithms $\mathsf{RangeProve}$ and $\mathsf{RangeVerify}$.

Inputs

- From Step 1: user holds Cred $= (C, \sigma, M)$ and secrets (b, r).
- Verifier knows the Authority public key pk_A and the current verification date T.

Protocol Run (What the User Sends to the Verifier)

1. **Verifier sets threshold.** The verifier computes the threshold:

$$t = \text{encode\_date}(T - 18 \text{ years}),$$

   which is public.

2. **User constructs a commitment to** $v = t - b$. Compute

$$D = g^t \cdot C^{-1} = g^{t-b} h^{-r}.$$

   Note that D is a Pedersen commitment to $v := t - b$ with randomness $-r$.

3. **User proves** $v \in [0, 2^k)$. The user runs a non-interactive range proof on D, showing that the committed value v lies in the interval:

$$\pi \leftarrow \mathsf{RangeProve}(D; \text{opening} = v, \text{rand} = -r, \text{bound} = 2^k).$$

   Fiat–Shamir is used inside the range-proof system, so π is non-interactive.

4. **User sends to verifier:**

$$\text{message} = (C,\, \sigma,\, M,\, D,\, \pi).$$

   Optionally, include a proof of possession of the credential (e.g., a PoK that the user knows the opening of C) if the verifier requires it. The range proof already implies knowledge of $(-r, v)$. A non-linking presentation nonce or session context may also be included.

Verifier Checks

1. **Signature check:** verify that σ is a valid signature by the Authority over (C, M), and check metadata M for expiry or revocation. Reject if invalid or revoked.

2. **Compute threshold:** recompute

$$t = \text{encode\_date}(T - 18 \text{ years}).$$

3. **Recompute/accept D:** recompute or verify that

$$D \stackrel{?}{=} g^t \cdot C^{-1}.$$

   This ensures that the user cannot substitute a different threshold.

4. **Range proof verification:** run

$$\mathsf{RangeVerify}(D, \pi, \text{bound} = 2^k).$$

   If it verifies, accept.

If all checks pass, the verifier is convinced that:

- The Authority attested to a credential binding some birthdate b to commitment C.
- The user has proven in zero knowledge that $b \leq t$ (i.e., age ≥ 18 as of T).

The verifier learns nothing else about b or any other user attribute.

5 Security and Implementation Notes

- Use secure elliptic-curve groups (e.g., Curve25519) and well-reviewed range-proof systems such as Bulletproofs.
- Ensure date encodings are bounded well below q to avoid modular wraparound.
- Include domain separation and nonces in all Fiat–Shamir transcripts to prevent replay attacks.
- Support revocation through short credential lifetimes or cryptographic accumulators.
- For unlinkability across multiple verifiers, use blind or anonymous credential systems (e.g., BBS+, Idemix, CL signatures).

Protocol Summary

Step	Operation
User	$C = g^b h^r$
Authority	$\sigma = \text{Sign}_{sk_A}(C, M)$
Verifier	Computes $t = \text{encode\_date}(T - 18 \text{ years})$
User	$D = g^t C^{-1}$
User	$\pi = \text{RangeProve}(D; v = t - b)$
User $\rightarrow$ Verifier	(C, σ, M, D, π)
Verifier	Checks σ, $D = g^t C^{-1}$, and $\text{RangeVerify}(D, \pi)$

6 Conclusion

The presented protocol allows a user to prove that they are over 18 years old without disclosing their exact birthdate. It combines the perfect hiding and additive homomorphism of Pedersen commitments with zero-knowledge range proofs, producing a minimal and privacy-preserving age-verification mechanism suitable for integration with digital ID systems.

7 Practical Cryptographic Recommendations

This section summarizes the main cryptographic and operational recommendations for implementing the proposed age-verification zero-knowledge protocol securely and efficiently. The goal is to ensure both privacy and integrity during credential issuance (Step 1) and proof verification (Step 2).

7.1 Group and Commitment Parameters

- **Group:** Use a prime-order elliptic-curve group, such as `secp256k1` or `Ed25519`, which offers efficient operations and well-audited libraries.
- **Generators:** Obtain independent generators g, h deterministically via a hash-to-curve function to avoid trapdoors. Ensure that no party knows $\log_g h$.
- **Commitments:** Pedersen commitments $C = g^m h^r$ should be computed using cryptographically secure randomness for r. These commitments provide both *hiding* and *binding* properties.

7.2 Authority Signature and Metadata

- **Digital Signature:** The issuing authority signs the hash of the commitment and credential metadata using a robust digital signature algorithm such as `Ed25519`, `ECDSA`, or `BLS`.
- **Metadata:** Include issuance time, expiry date, and a unique credential identifier within the signed data. This ensures verifiability and supports short-lived credentials.
- **Revocation:** Attach a revocation handle (e.g., the hash of a random revocation secret known to the authority) or maintain a revocation list/OCSP-like mechanism. Short-lived credentials can reduce the need for complex revocation checks.

7.3 User Binding and Replay Prevention

- **Binding to User:** To prevent credential replay by unauthorized users, bind the credential to the user's public key PK_U. The authority should sign the combined hash:

  $$\mathrm{Sign}_{\mathrm{Authority}}\big(H(C \| PK_U \| M)\big)$$

  where M represents additional metadata. This approach slightly reduces unlinkability but increases theft-resistance.
- **Proof-of-Possession:** The authority must verify the user's real-world identity and birth date during issuance. The user must demonstrate in zero-knowledge that the committed birth date used in the proof matches the one verified by the authority.

7.4 Nonce and Expiry Management

- **Nonces:** Use nonces in challenge–response steps to prevent replay attacks in the zero-knowledge proof phase.

- **Expiry:** Credentials should include short validity periods to minimize the risk from compromised commitments or private keys.

7.5 Implementation and Verification Tools

- **Formal Verification:** Use formal analysis tools such as `ProVerif`, `Tamarin`, or `F*` to verify key security properties including:
  - *Soundness:* Only users with valid commitments can pass verification.
  - *Zero-Knowledge:* No information about the user's actual age is revealed beyond the statement "age > 18".
  - *Unforgeability:* The credential cannot be fabricated or reused by others.

7.6 Summary of Design Properties

The design ensures:

- **Privacy:** No personal data (name, exact birthdate) is disclosed.

- **Integrity:** The verifier confirms the credential's authenticity and issuance by a trusted authority.

- **Unlinkability:** Different proofs by the same user cannot be linked, provided r and proof randomness are unique each time.

- **Non-Forgeability:** Without the private key or correct commitment randomness, generating a valid proof is infeasible.

8 To the Next Level: Advanced Design Enhancements

Beyond the foundational protocol and basic security recommendations, several refinements can elevate the age-verification system from a conceptual prototype to a production-grade privacy infrastructure. These address practical deployment, interoperability, and resistance to subtle cryptographic pitfalls.

8.1 Elliptic Curve and Parameter Hardening

- **Curve Choice:** Adopt a modern, secure elliptic curve such as `Curve25519` or `secp256k1`, depending on compatibility requirements. These curves are well-studied and have efficient implementations in multiple languages.

- **Generator Derivation:** Derive generators g, h deterministically using a *hash-to-curve* function to eliminate trust assumptions. Ensure that no entity can compute $\log_g h$.

- **Encoding and Domain Parameters:** Choose encoding and field parameters so committed values remain small and avoid modular wrap-around when used in range or comparison proofs.

8.2 Advanced Zero-Knowledge Components

- **Range Proofs:** Integrate a well-audited range proof system such as `Bulletproofs` or `LegoSNARKs` to ensure committed ages are non-negative and within realistic bounds.
- **Comparison Proofs:** Optimize the "age > 18" comparison using modular decomposition (binary representation) or constraint systems compatible with the chosen ZK backend.

8.3 Revocation and Credential Lifecycle

- **Revocation Strategy:** Define a robust revocation mechanism. Options include:
  1. Time-bounded credentials with short expiry.
  2. Revocation lists or status servers (OCSP-like).
  3. Cryptographic accumulators for scalable privacy-preserving revocation.
- **Expiry Policy:** Encourage periodic credential renewal to mitigate key compromise or metadata leakage.

8.4 Privacy and Linkability Policy

- **Anonymous vs. Linkable Credentials:** Decide early whether the system should allow multiple proofs from the same credential to be unlinkable.
- **Techniques:** Employ credential blinding or anonymous credentials (e.g., based on BBS+ or CL signatures) if unlinkability is a requirement. For regulated environments, controlled linkability may be acceptable.

8.5 Hash Domain Separation and Context Binding

- **Domain Separation:** Introduce unique prefixes or tags in every hash used in Fiat–Shamir transformations or commitment derivations to avoid cross-protocol replay or collision with other systems.
- **Context Binding:** Include protocol version, service domain, and verifier identity within the hash transcript to ensure proofs are valid only in their intended context.

8.6 Integration and Security Verification

- **Formal Verification:** Model the full protocol in `ProVerif`, `Tamarin`, or `F` to analyze confidentiality, soundness, and unlinkability.
- **Implementation Review:** Conduct independent audits focusing on randomness generation, nonce uniqueness, and memory safety in proof libraries.
- **Interoperability:** Use standardized serialization formats (CBOR, JSON-LD, or DID-compatible structures) to integrate with digital identity frameworks such as W3C Verifiable Credentials.

8.7 Outcome

Incorporating these enhancements yields a protocol that is:

- Cryptographically hardened against subtle parameter misuse.
- Extensible to richer ZK credential ecosystems.
- Compatible with revocation, policy, and interoperability layers.
- Securely bound to its application context and resistant to replay or linkage attacks.

9 Diagram

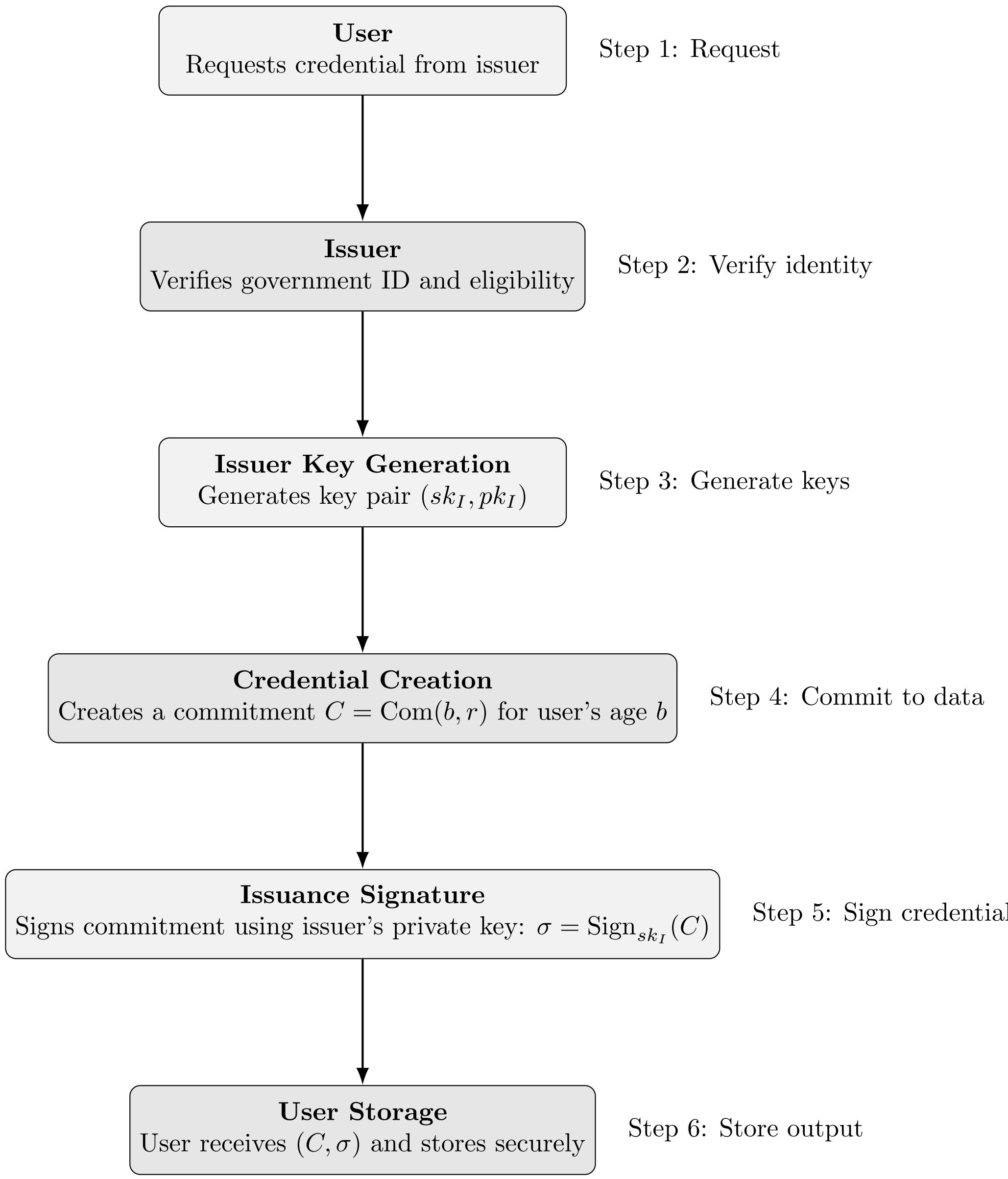

Figure 1: Step 2 – Getting Digital ID using Pedersen Commitments

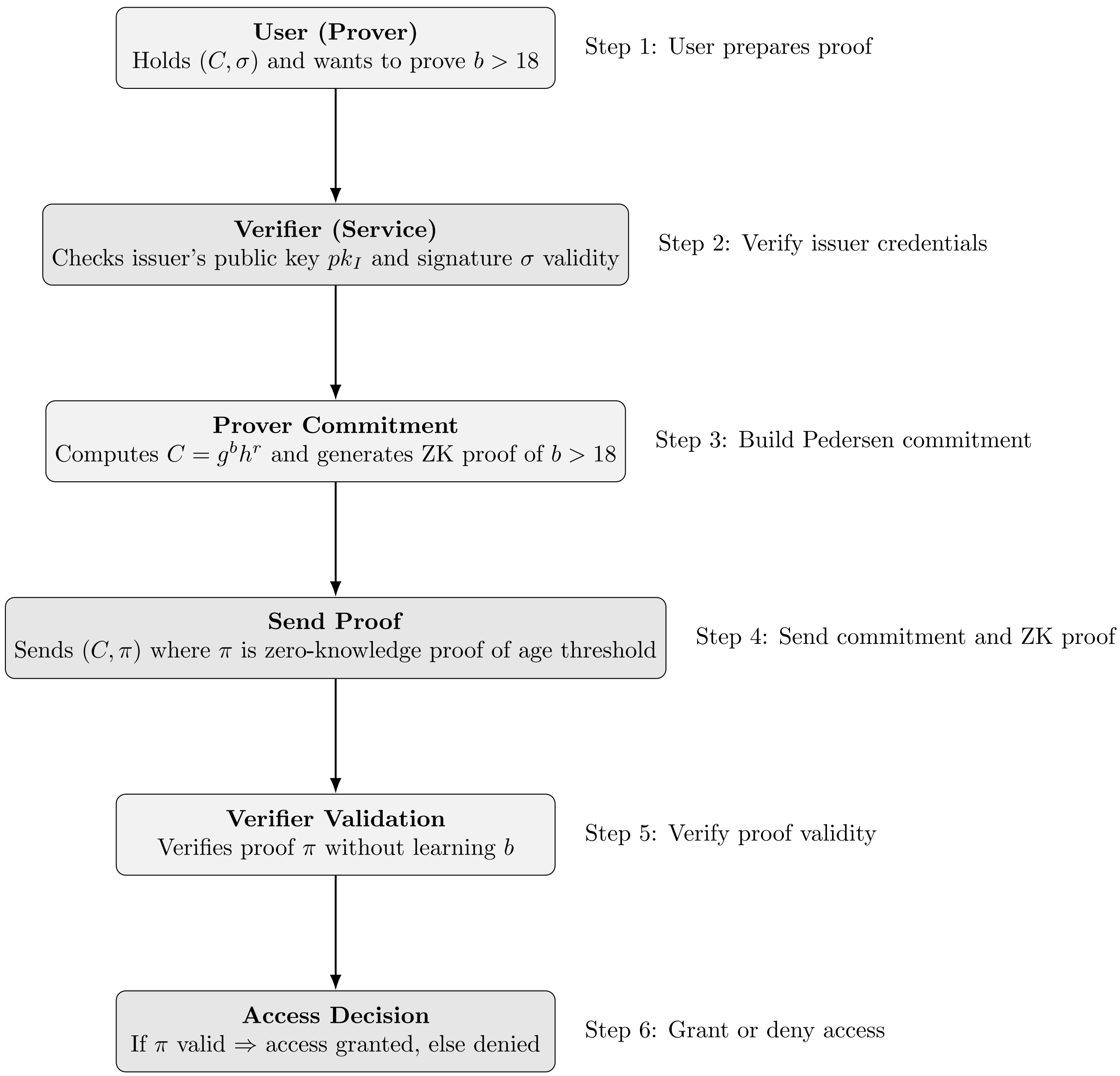

Figure 2: Step 2 – Zero-Knowledge Age Verification Protocol using Pedersen Commitments