Zero-Knowledge Age Verification Protocol Using Pedersen Commitments

—

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.

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 trusted authority, together with identifying information (in a secure and authenticated channel).

3) The authority verifies the user's real-world ID and signs the commitment:

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

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

4) The authority returns the credential:

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

3.3 Properties

- The authority attests that C commits to the correct birthdate, but cannot later read b.
- The user can prove statements about b without revealing it, using zero-knowledge proofs.

4 Step 2: Zero-Knowledge Age Verification

4.1 Objective

Given (C, σ, M) and secret opening (b, r), the user wants to convince a verifier that their age is greater than 18 without revealing b.

4.2 Public Parameters

- Group (G, q, g, h) as above.
- Authority's public key pk_A for verifying σ.
- Date encoding function encode_date$(\cdot)$.

4.3 Protocol

1) **Threshold computation:** The verifier computes

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

where T is today's date.

2) **Construct commitment for the difference:** The user computes

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

Let $v = t - b$. Then D is a Pedersen commitment to v with randomness $-r$.

3) **Prove non-negativity:** The user proves in zero knowledge that $v \in [0, 2^k)$ for some suitable bound 2^k. This ensures $t - b \geq 0$, i.e., $b \leq t$.

The proof is a non-interactive *range proof*:

$$\pi \leftarrow \mathrm{RangeProve}(D; v, -r, 2^k).$$

4) **Send to verifier:** The user sends (C, σ, M, D, π).

5) **Verifier checks:**

   a) Verify the authority signature σ on (C, M).
   b) Recompute t and verify $D = g^t C^{-1}$.
   c) Verify the range proof π using $\mathrm{RangeVerify}(D, \pi, 2^k)$.
   d) Optionally check revocation/expiry in M.

If all checks pass, the verifier is convinced that:

- The credential was issued by a legitimate authority.
- The hidden birthdate b satisfies $b \leq t$ (i.e., age ≥ 18).
- No additional personal information is revealed.

4.4 Soundness and Zero-Knowledge

- **Soundness:** The verifier is convinced that $b \leq t$ only if the range proof is valid. Since the commitment is binding, the user cannot fake a younger age.
- **Zero-knowledge:** Pedersen commitments and range proofs reveal nothing about b or r. The verifier learns only that the age exceeds the threshold.

4.5 Numeric Example (Illustrative Only)

Let the group be modulo $p = 23$ with generators $g = 2$, $h = 5$ (toy parameters).

$$\begin{aligned}
T &= 100, \quad t = 82, \\
b &= 80, \quad r = 3, \\
C &= g^b h^r = 2^{80} \cdot 5^3 \pmod{23}, \\
D &= g^t C^{-1} = g^2 h^{-3} = g^{t-b} h^{-r}.
\end{aligned}$$

Here $v = t - b = 2$. A range proof shows that $v \in [0, 2^8)$, proving $b \leq t$ (age ≥ 18).

4.6 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).

4.7 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 $\to$ Verifier	(C, σ, M, D, π)
Verifier	Checks σ, $D = g^t C^{-1}$, and $\text{RangeVerify}(D, \pi)$

5 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.

6 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).

6.1 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.

6.2 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.

6.3 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.

6.4 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.

6.5 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.

6.6 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.

7 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.

7.1 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.

7.2 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.

7.3 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.

7.4 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.

7.5 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.

7.6 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.

7.7 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.