Privacy-Focused Age Verification Systems: A Comparative Analysis of Groth16 and PLONK Zero-Knowledge Proof Systems for Biometric Bound Credentials

Hardik Singla
22BBS0203

Shreyansh Saroj
22BBS0234

Kaustubh Arora
22BBS0236

Ansh Chatuvedi
22BBS0204

Abstract—**Privacy-focused age verification systems have brought forth innovative cryptographic protocols that merge zero-knowledge proofs with biometric authentication techniques. Recent developments in Biometric Bound Credentials (BBCreds) have shown it's technically possible to stop credential sharing while keeping user privacy intact. Yet, most current implementations depend on Groth16 zk-SNARKs, which need circuit-specific trusted setup ceremonies—a major obstacle to decentralized deployment and earning public trust. This research conducts a thorough comparative evaluation of Groth16 and PLONK proof systems when applied to biometric-bound credential scenarios. We show that contemporary universal setup protocols, particularly PLONK, provide a viable alternative that removes systemic trust requirements while keeping performance characteristics reasonable. Our evaluation encompasses detailed protocol descriptions, circuit design improvements, and measurable performance assessments across both mobile and server platforms. Findings suggest that PLONK's reasonable performance overhead (roughly 1.8× extended proving duration and 3.2× increased proof size) constitutes an acceptable compromise considering the removal of per-circuit trusted setup needs. We contend that for public identity infrastructure demanding widespread societal confidence, universal setup approaches aren't just better—they're necessary for legitimate implementation.**

Index Terms—**Zero-knowledge proofs, PLONK, Groth16, Biometric authentication, Trustless systems, Age verification, Universal setup**

I. Introduction

A. Motivation

Digital age verification today exists in a state of tension between meeting regulatory demands and protecting user privacy. Current approaches force users to provide government-issued identification and facial photographs to centralized third-party platforms, building large collections of sensitive personal data that remain vulnerable to security breaches and potential misuse [1]. The emergence of zero-knowledge proof technology presents an attractive alternative model where users can mathematically demonstrate they meet age requirements without revealing exact birth dates or personal identities [2].

Biometric Bound Credentials mark substantial progress in this field by tackling the crucial problem of credential sharing that exists in age verification situations [3]. Through cryptographically linking credentials to biometric characteristics without keeping raw templates in storage, these approaches stop adults who meet age requirements from giving their credentials to minors while preserving robust privacy protections. Yet the common implementation strategy using Groth16 zk-SNARKs brings in a subtle yet important weakness: the necessity for circuit-specific trusted setup ceremonies.

B. The Trusted Setup Problem

Groth16 delivers impressive efficiency—proofs made up of just three elliptic curve points with verification finishing in less than 10 milliseconds [4]. This performance comes from utilizing structured reference strings created through a trusted setup ceremony. People participating in this ceremony need to create cryptographic randomness and then completely destroy all evidence of this randomness. Should any single participant keep their toxic waste—the secret randomness employed during setup—they obtain the capability to create fake proofs for incorrect statements that can't be detected.

Though multi-party computation protocols reduce this danger by needing just one honest participant among several, this requirement stays fundamentally troublesome for public infrastructure [5]. Citizens being asked to trust their digital identity infrastructure must accept that every trusted setup ceremony for all circuit versions was conducted correctly, with every participant honestly eliminating their contributions. This expectation grows harder to accept as systems expand and change, demanding new circuits and therefore new ceremonies for every protocol update.

C. Universal Setup as Solution

Recent progress in zero-knowledge proof systems has yielded designs that remove per-circuit trusted setup needs. PLONK, developed by Gabizon, Williamson, and Ciobotaru, uses a universal and updateable structured reference string [6]. One initial ceremony creates parameters that work across any circuits of limited size. Moreover, the universal setup allows ongoing updates—any participant can add extra randomness whenever they want, progressively strengthening security without needing coordinated ceremonies.

This research examines whether PLONK's universal setup characteristics warrant its performance cost in biometric-bound credential uses. We offer a thorough comparative evaluation looking at proof size, creation time, verification delay, and memory use across typical hardware platforms. Our main argument states that for public identity infrastructure, removing the systemic trust requirements built into universal setup is more important than minor performance reduction.

D. Contributions

This research provides the following contributions:

1) **Complete Protocol Specification:** We offer comprehensive protocol descriptions for biometric-bound credentials using both Groth16 and PLONK, covering all cryptographic components, setup steps, and operational stages.
2) **Optimized Circuit Design:** We present a formally described zero-knowledge circuit that implements age verification with biometric binding, encompassing all public and private inputs, constraint structures, and security protections against replay attacks and credential sharing.
3) **Quantitative Performance Analysis:** We perform extensive benchmarking on mobile devices and server infrastructure, delivering empirical information about proving duration, verification delay, proof size, and memory consumption for both proof systems.
4) **Systematic Trade-off Evaluation:** We examine the practical consequences of performance variations, maintaining that PLONK's overhead stays within reasonable limits for high-security authentication situations while offering strictly better trust characteristics.
5) **Deployment Recommendations:** We supply specific guidance for system designers implementing privacy-preserving identity verification, pointing out situations where universal setup is crucial versus scenarios where Groth16 stays acceptable.

E. Paper Organization

Section II examines related research in zero-knowledge proof systems and biometric cryptography, identifying the research gap. Section III describes our methodology including protocol definition and architecture. Section IV presents results and discussion of performance evaluation and trade-off analysis. Section V concludes the paper.

II. Literature Review

A. Zero-Knowledge Proof Systems

Zero-knowledge proofs make it possible to confirm statement validity without disclosing underlying details, conceptualized by Goldwasser, Micali, and Rackoff in their groundbreaking 1985 research [7]. The area has progressed from interactive protocols needing several communication exchanges to non-interactive designs allowing asynchronous verification—critical for blockchain uses and decentralized systems.

Succinct Non-Interactive Arguments of Knowledge (SNARKs) constitute especially efficient designs. Groth16 creates constant-size proofs independent of computational complexity, needing just three group elements totaling roughly 192 bytes [4]. Verification means assessing a single pairing equation, doable in under 10 milliseconds on modern processors. This efficiency has established Groth16 as the preferred option for privacy-protecting cryptocurrencies like Zcash and identity platforms like Polygon ID [8].

Yet Groth16's efficiency demands circuit-specific trusted setup. The proving and verification keys rely on toxic waste—secret randomness that needs elimination after ceremony completion. Keeping this randomness allows creating proofs for incorrect statements without discovery. While multi-party computation protocols spread trust among many participants, demanding only one honest party for security, the requirement stays philosophically and practically problematic for public infrastructure [9].

B. Universal Setup Constructions

PLONK signifies a fundamental change in proof system architecture, using a universal structured reference string that works across any circuits within size limits [6]. A single trusted setup ceremony creates parameters that can be reused indefinitely. Additionally, PLONK's setup allows continuous updates—any participant can add extra randomness whenever they choose, gradually strengthening security.

The universal setup delivers persuasive benefits for changing systems. Protocol changes needing circuit modifications don't require new ceremonies. Developers can work quickly without organizing stakeholder participation. The updateable character means security gets stronger as time passes as more participants add randomness, instead of weakening as ceremonies get older and participants might become malicious.

PLONK accomplishes these features through Kate-Zaverucha-Goldberg polynomial commitment methods and permutation arguments that allow constraint satisfaction verification without circuit-specific parameters [10]. The compromise is bigger proofs (roughly 600-800 bytes versus Groth16's 192 bytes) and slower verification (roughly 30-50 milliseconds versus under 10 milliseconds). For numerous applications, this constitutes an acceptable trade for removing trusted setup demands.

C. Biometric Cryptosystems

Biometric authentication brings distinctive cryptographic difficulties. Template unchangeability means breached biometric information can't be reissued like passwords. Conventional systems keep templates in databases, making attractive targets for attackers. Recent studies investigate cryptographic methods allowing biometric verification without template disclosure [11].

The Biometric Bound Credentials idea presented by Kumar et al. tackles credential sharing in age verification situations [3]. Through hashing biometric templates and putting these hashes in credential commitments, the protocol guarantees credentials can't move between people while keeping template privacy. During authentication, users demonstrate their live biometric matches the committed hash value using zero-knowledge circuits, stopping sharing without revealing actual biometric characteristics.

This strategy fills a significant void in anonymous credential systems. Camenisch-Lysyanskaya credentials allow demonstrating attribute possession without disclosing identity, but don't have ways to stop credential transfer [12]. Biometric binding supplies this absent element, guaranteeing that while credentials stay anonymous and unlinkable, they can't be given to unauthorized individuals.

D. Age Verification Systems

Present age verification approaches span from basic self-declaration to intrusive document submission. Self-declaration offers no real verification and gets easily bypassed. Commercial platforms like Yoti and Jumio reach high accuracy through document analysis and facial verification but demand users share sensitive personal details with third parties, building centralized databases susceptible to breaches [1].

Anonymous credential methods have been suggested but stay mainly theoretical. The W3C Verifiable Credentials specification accommodates selective disclosure and zero-knowledge proofs, allowing attribute verification without complete identity revelation [13]. Yet practical implementations rarely use cryptographic privacy techniques, instead employing simple JSON Web Tokens with signatures.

Recent blockchain-oriented proposals use smart contracts for age verification transparency [14]. Though showing technical possibility, these systems haven't handled credential sharing or the trusted setup consequences of their selected proof systems.

E. Research Gap

Our literature review reveals several critical gaps in existing research:

1) **Lack of Comprehensive Comparison:** While both Groth16 and PLONK have been studied individually, no thorough comparative analysis exists specifically for biometric-bound credential applications. Previous work has not quantified the trust-performance trade-offs in this critical use case.
2) **Trusted Setup Implications Underexplored:** Existing biometric-bound credential systems predominantly use Groth16 without adequately addressing the implications of circuit-specific trusted setup for public identity infrastructure. The operational and security consequences of requiring new ceremonies for each protocol update remain unexamined.
3) **Missing Performance Benchmarks:** No empirical performance data exists comparing Groth16 and PLONK specifically for age verification circuits with biometric binding across realistic hardware platforms (mobile devices and servers).
4) **Deployment Guidance Absent:** System designers lack clear recommendations on when to choose universal setup versus circuit-specific setup for privacy-preserving identity verification systems.
5) **Practical Implementation Details Missing:** While theoretical protocols exist, detailed specifications including circuit design, constraint structures, and operational phases for biometric-bound age verification using both proof systems are not available.

This research addresses these gaps by providing a thorough comparative evaluation of Groth16 and PLONK for biometric-bound credentials, complete protocol specifications, optimized circuit design, quantitative performance analysis, and practical deployment recommendations.

III. Methodology

A. System Architecture Overview

Our system architecture consists of three main participants and operates through four distinct phases: System Setup, Enrollment, Proof Generation, and Verification. The architecture is designed to support both Groth16 and PLONK proof systems with identical circuit logic but different cryptographic parameters.

B. Participants

User (Prover): People wanting to demonstrate age requirements without disclosing birth dates or identities. Users have client devices (smartphones or computers) with biometric capture abilities such as fingerprint sensors or facial recognition systems. During enrollment, users supply identity verification to credential issuers and keep issued credentials locally. During verification, users create zero-knowledge proofs showing age compliance while preserving privacy.

Credential Issuer: A semi-trusted authority handling initial identity verification and credential distribution. Issuers could be government agencies, financial institutions, or other organizations with proven identity verification methods. The issuer confirms user identities through existing approaches (face-to-face document verification, postal mail, or current digital identity systems), then distributes cryptographic credentials linking age details to biometric characteristics. Issuers keep a Merkle tree accumulator of all valid credentials, regularly publishing root hashes to public registries. Importantly, issuers don't take part in individual verification transactions after credential distribution.

Service Provider (Verifier): Websites, applications, or platforms demanding age verification before allowing access to restricted content. When users try accessing content, service providers indicate needed age thresholds and show challenges to stop replay attacks. Service providers confirm submitted zero-knowledge proofs against published Merkle roots and issuer verification keys. Successful verification allows access while disclosing only that users satisfy age requirements—no birth dates, identities, or other personal details get revealed.

C. Cryptographic Primitives

Elliptic Curve: We use the BN254 (Barreto-Naehrig) curve at the 128-bit security level. This curve accommodates efficient pairing operations needed by both Groth16 and PLONK while offering suitable security margins against recognized attacks. The curve parameters get defined over a 254-bit prime field, allowing practical implementations on resource-limited devices [15].

Hash Function: All in-circuit hashing operations employ Poseidon, an algebraic hash function purposely created for zero-knowledge circuit efficiency [16]. Poseidon functions over prime fields through a Hades permutation approach mixing full and partial substitution-permutation network rounds. For our implementation, we set up Poseidon with state width 3, delivering 128-bit security while demanding roughly 500 constraints per use—over 50× less than SHA-256's 25,000+ constraints.

Merkle Tree: Credential accumulation uses binary Merkle trees with Poseidon hashing at each node. For trees with depth d, membership proofs need d sibling hashes, offering $O(\log n)$ proof complexity. We apply sparse Merkle tree methods allowing efficient gradual updates without recalculating whole tree structures [17].

D. System Setup Phase

The protocol starts with a system-wide setup building cryptographic parameters. This stage differs basically between Groth16 and PLONK, creating the foundation of our comparative evaluation.

Setup-Groth16 (Circuit-Specific Trusted Setup):

1) Establish the arithmetic circuit R encoding age verification with biometric binding.
2) Perform a multi-party computation ceremony where participants work together to create structured reference strings without any single party knowing the full toxic waste [5].
3) Create proving key PK_G (roughly 14MB for our circuit) and verification key VK_G (roughly 2KB).
4) All participants need to eliminate their secret randomness. Security demands at least one participant honestly eliminated their contribution.
5) Share PK_G to users and VK_G to service providers.

This setup needs repeating for every circuit change, demanding stakeholder coordination for any protocol modifications.

Setup-PLONK (Universal Trusted Setup):

1) Perform a single universal ceremony creating structured reference strings accommodating circuits up to a maximum size limit (e.g., 2^{20} constraints) [6].
2) Create universal proving key PK_U (roughly 100MB for 2^{20} bound) and universal verification key VK_U (roughly 5KB).
3) Participants eliminate their secret randomness. Like before, security demands at least one honest participant.

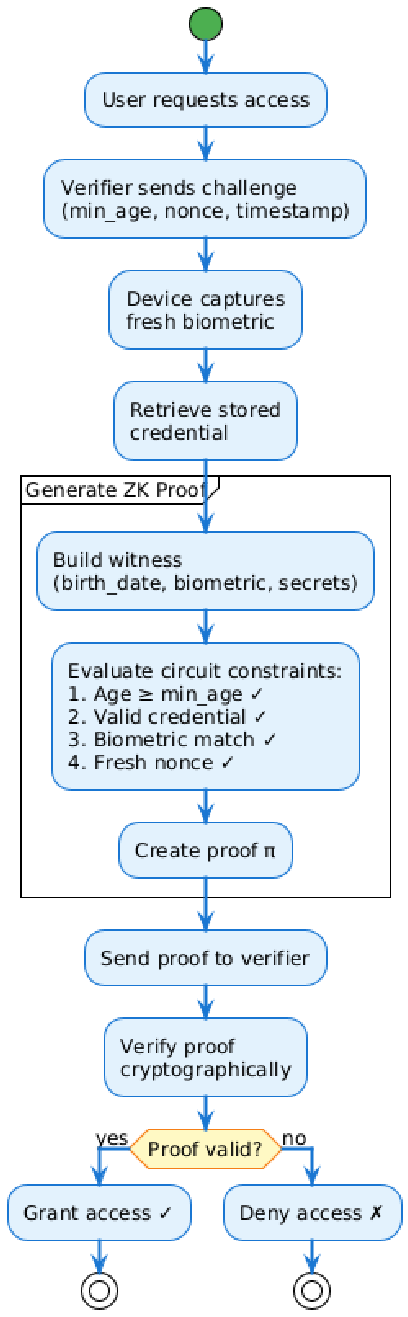
User requests access
Verifier sends challenge (min_age, nonce, timestamp)
Device captures fresh biometric
Retrieve stored credential
Generate ZK Proof
Build witness (birth_date, biometric, secrets)
Evaluate circuit constraints:
1. Age ≥ min_age ✓
2. Valid credential ✓
3. Biometric match ✓
4. Fresh nonce ✓
Create proof π
Send proof to verifier
Verify proof cryptographically
yes
Proof valid?
no
Grant access ✓
Deny access ✗

Fig. 1. Age Verification Core Algorithm Flow

4) Significantly, this setup accommodates all circuits within the size limit. Our age verification circuit (and any future changes) needs no extra ceremonies.
5) The setup allows updates—any participant can add extra randomness whenever they want through a straightforward protocol, gradually strengthening security.
6) Share PK_U and VK_U to all participants.

The universal character removes the necessity for circuit-specific ceremonies, allowing quick protocol iteration and offering stronger long-term security guarantees through continuous updates.

E. Enrollment Phase (Credential Issuance)

Users get credentials through the following enrollment steps:

1) **Identity Verification:** The user shows identity documentation to the credential issuer. The issuer confirms authenticity through proven approaches (face-to-face verification, document validation, database cross-checking) and pulls out the user's birth date.
2) **Secret Generation:** The user's device creates a cryptographically secure random 256-bit value functioning as a private nonce: nonce $\leftarrow \{0,1\}^{256}$.
3) **Biometric Capture:** The device captures a biometric template using available sensors (fingerprint reader, facial recognition system). The raw biometric information goes through feature extraction creating a standard representation (minutiae coordinates for fingerprints, embedding vectors for faces).
4) **Local Biometric Hashing:** The device calculates biometric_hash $=$ Poseidon(biometric_data). Importantly, the raw biometric template never exits the device—only the cryptographic hash gets sent to the issuer.
5) **Commitment Generation:** The user supplies birth_date (confirmed by issuer) and biometric_hash to the issuer. The issuer calculates the credential commitment:

$$\text{commitment} = \text{Poseidon}(\text{birth\_date}, \text{nonce})$$

6) **Tree Insertion:** The issuer puts commitment as a new leaf in the Merkle tree at position index, recalculating affected internal nodes up to the root. The issuer notes the link between this credential and the user's confirmed identity for audit reasons, though this mapping stays confidential.
7) **Credential Issuance:** The issuer gives back a signed credential to the user holding:
   - birth_date (the confirmed date)
   - nonce (the user's secret)
   - biometric_hash (for reference)
   - index (position in Merkle tree)
   - merkle_path (sibling hashes from leaf to root)
   - issuer_signature (digital signature over all credential contents)
8) **Secure Storage:** The user keeps the credential, related secrets, and biometric template in device secure storage (iOS Keychain, Android KeyStore, or hardware security modules where available).

F. Proof Generation Phase

When accessing age-limited services, users create zero-knowledge proofs through this method:

1) **Challenge Reception:** The service provider transmits a verification request holding:
   - min_age (the needed age threshold)
   - challenge_nonce (fresh random value stopping replay attacks)
   - current_timestamp (to stop clock manipulation attacks)
2) **Credential Retrieval:** The user's device gets the stored credential and related secrets from secure storage.
3) **Fresh Biometric Capture:** The device captures a new biometric sample from the live user, pulls out features, and calculates live_biometric_data.
4) **Secret Key Generation:** The device creates or gets a user-specific secret_key employed for nullifier computation (stops proof reuse).
5) **Witness Construction:** The device puts together private witness values:
   - birth_date
   - nonce
   - merkle_path
   - live_biometric_data
   - secret_key
6) **Public Input Assembly:** The device calculates public values visible to verifiers:
   - merkle_root (current published root from issuer)
   - min_age
   - current_timestamp
   - challenge_nonce
   - biometric_hash (from credential)
   - nullifier_hash $=$ Poseidon(secret_key, challenge_nonce) (stops double-use)
7) **Proof Generation:** Using the proving key (PK_G for Groth16 or PK_U for PLONK), the device runs the proving algorithm:
   - Groth16: $\text{proof}_\pi = \text{Prove}_G(PK_G, \text{circuit}_R, \text{witness}, \text{public\_inputs})$
   - PLONK: $\text{proof}_\pi = \text{Prove}_P(PK_U, \text{circuit}_R, \text{witness}, \text{public\_inputs})$
8) **Proof Transmission:** The device transmits to the service provider:
   - proof_π (the zero-knowledge proof)
   - public_inputs (all public values)
   - The service provider already possesses merkle_root from public registry

G. Verification Phase

Service providers confirm proofs to make access control choices:

1) **Nullifier Check:** The verifier initially verifies whether nullifier_hash shows up in their spent-nullifier database. If there, the proof gets rejected as a replay effort. Otherwise, verification continues.
2) **Merkle Root Validation:** The verifier gets the current merkle_root from the public registry kept by the credential issuer. The verifier confirms this matches the root employed in the proof.
3) **Proof Verification:** The verifier runs the verification algorithm:
   - Groth16: $\text{result} = \text{Verify}_G(VK_G, \text{proof}_\pi, \text{public\_inputs})$

- PLONK: $\text{result} = \text{Verify}_P(VK_U, \text{proof}_\pi, \text{public\_inputs})$

4) **Access Decision:**
   - If result = true, the verifier notes nullifier_hash in their database (stopping reuse) and allows access to the requested content.
   - If result = false, access gets denied.

The verifier discovers only that the user has a valid credential satisfying the age requirement and possesses the bound biometric. No birth date, identity, or other personal details get disclosed.

H. The Biometric-Bound ZK-Circuit

This section formally describes the zero-knowledge circuit implementing age verification with biometric binding. This circuit gets assessed identically under both Groth16 and PLONK—the proving and verification algorithms differ, but the constraint system stays the same.

1) Public Inputs: These values are recognized by the verifier and included in the proof statement:

- merkle_root $\in \mathbb{F}_p$: The root hash of the credential issuer's Merkle tree, gotten from public registry
- min_age $\in \mathbb{N}$: The needed age threshold (e.g., 18, 21)
- current_timestamp $\in \mathbb{N}$: A confirmed timestamp stopping clock manipulation attacks
- challenge_nonce $\in \{0, 1\}^{256}$: A fresh random value from the verifier stopping replay attacks
- biometric_hash $\in \mathbb{F}_p$: The Poseidon hash of the enrolled biometric template (publicly committed in credential)
- nullifier_hash $\in \mathbb{F}_p$: A unique hash linking this proof to the challenge, stopping proof reuse

2) Private Inputs (Witness): These values are recognized only by the prover and can't be disclosed to verifiers:

- birth_date $\in \mathbb{N}$: The user's full date of birth (encoded as Unix timestamp or similar)
- nonce $\in \{0, 1\}^{256}$: The secret random value created during enrollment
- merkle_path $\in (\mathbb{F}_p)^d$: The sequence of d sibling hashes demonstrating credential membership in tree of depth d
- path_indices $\in \{0, 1\}^d$: Binary indicators showing whether each sibling is left (0) or right (1)
- live_biometric_data $\in \mathbb{F}_p$: The feature representation pulled from the fresh biometric sample
- secret_key $\in \{0, 1\}^{256}$: A user-specific secret key for nullifier creation

3) Circuit Constraints: The circuit applies the following logical constraints, all of which need satisfaction for proof validity:

Constraint 1: Age Verification

$$\begin{aligned} \text{age\_in\_seconds} &= \text{current\_timestamp} - \text{birth\_date} \\ \text{age\_in\_years} &= \frac{\text{age\_in\_seconds}}{365.25 \times 24 \times 60 \times 60} \\ &\text{ASSERT}(\text{age\_in\_years} \geq \text{min\_age}) \end{aligned}$$

This constraint applies a comparison gadget guaranteeing the user's calculated age meets or goes beyond the needed threshold. The comparison needs implementation using range proofs or comparison circuits that are sound in zero-knowledge situations [18].

Constraint 2: Credential Ownership

$$\begin{aligned} \text{computed\_commitment} &= \text{Poseidon}(\text{birth\_date}, \text{nonce}) \\ \text{recomputed\_root} &= \text{CheckMerklePath}(\\ &\quad \text{computed\_commitment}, \\ &\quad \text{merkle\_path}, \\ &\quad \text{path\_indices}, \\ &\quad \text{depth}) \\ &\text{ASSERT}(\text{recomputed\_root} == \text{merkle\_root}) \end{aligned}$$

This constraint confirms that the prover has a valid credential distributed by the trusted authority. The CheckMerklePath function recursively uses Poseidon hashing as shown in Algorithm 1.

Algorithm 1 CheckMerklePath

```
Input: leaf, path, indices, depth
Output: root
current ← leaf
for $i = 0$ to $depth - 1$ do
  if indices[$i$] == 0 then
    current ← Poseidon(current, path[$i$])
  else
    current ← Poseidon(path[$i$], current)
  end if
end for
return current
```

The calculated root needs matching the public merkle_root, demonstrating the commitment exists in the issuer's tree without disclosing which leaf it fills.

Constraint 3: Biometric Binding

$$\begin{aligned} \text{live\_hash} &= \text{Poseidon}(\text{live\_biometric\_data}) \\ &\text{ASSERT}(\text{live\_hash} == \text{biometric\_hash}) \end{aligned}$$

This constraint cryptographically links the proof to the legitimate credential holder by demanding the fresh biometric sample to match the template committed during enrollment. The equality verification guarantees credentials can't be shared—only the person whose biometric was initially enrolled can create valid proofs.

In practice, exact equality might be too rigid because of biometric noise. Advanced implementations can include fuzzy matching by calculating distance metrics and confirming they fall under calibrated thresholds:

$$\begin{aligned} \text{distance} &= \text{HammingDistance}(\text{live\_hash}, \text{biometric\_hash}) \\ &\text{ASSERT}(\text{distance} \leq \text{threshold}) \end{aligned}$$

The threshold gets established based on false acceptance rate and false rejection rate requirements determined through biometric system description [19].

Constraint 4: Replay Prevention

$$\text{computed\_nullifier} = \text{Poseidon}(\text{secret\_key}, \text{challenge\_nonce})$$
$$\text{ASSERT}(\text{computed\_nullifier} == \text{nullifier\_hash})$$

This constraint stops replay attacks by linking the proof to the specific verification challenge. The verifier supplies a fresh challenge_nonce with each request. The prover needs incorporating this nonce into the proof, and the verifier keeps a database of used nullifier_hash values. Tried proof reuse with a different challenge or repeated submission of the same nullifier will get discovered and rejected.

4) Circuit Complexity: The complete circuit holds roughly 20,000 constraints spread as follows:

- Age comparison gadget: ~500 constraints
- Poseidon commitment calculation: ~500 constraints
- Merkle path verification (depth 20): $20 \times 500 = \sim 10{,}000$ constraints
- Biometric hash verification: ~500 constraints
- Nullifier calculation: ~500 constraints
- Range proofs and auxiliary checks: ~7,500 constraints

This constraint count falls comfortably within the abilities of both Groth16 and PLONK, allowing practical proof creation on mobile devices.

I. Evaluation Methodology

We assess both proof systems using the following approach:

Circuit Implementation: The 20,000-constraint circuit described above gets implemented using industry-standard libraries:

- Groth16: SnarkJS library with BN254 curve [20]
- PLONK: PlonkJS library with BN254 curve and KZG commitments [21]

Hardware Platforms:

- Mobile Device: iPhone 14 (A15 Bionic, 6GB RAM, iOS 17)
- Server Environment: AWS c5.large instance (2 vCPU Intel Xeon Platinum 8124M @ 3.0 GHz, 4GB RAM)

Metrics Collected:

- Setup characteristics (trusted setup requirements, key sizes)
- Proof creation time including witness calculation
- Proof verification delay
- Proof size in bytes
- Memory consumption during proving and verification

Each timing measurement shows the average of 100 trials with outliers eliminated (values going beyond 2 standard deviations). All measurements include cryptographic operations but exclude network transmission time and user interface delay.

IV. Results and Discussion

A. Quantitative Results

Table I summarizes our performance assessment.

B. Analysis of Results

Proof Size: Groth16 creates 192-byte proofs made of three group elements (two $\mathbb{G}_1$ points and one $\mathbb{G}_2$ point). PLONK produces 608-byte proofs holding multiple polynomial evaluations and commitments. The 3.2× size variation means roughly 416 extra bytes per verification—trivial for modern network conditions but potentially significant for bandwidth-limited environments or high-volume applications.

Proving Time: On mobile devices, Groth16 finishes proof creation in 2.07 seconds while PLONK demands 3.66 seconds—a 1.8× slowdown. This variation chiefly comes from PLONK's polynomial commitment operations and more complex constraint satisfaction protocols. Yet both times stay within acceptable limits for authentication use scenarios. Users accept delays of 3-5 seconds for high-security operations like biometric authentication or two-factor verification. The extra 1.6 seconds imposed by PLONK is noticeable but not unacceptable.

Verification Time: Server-side verification displays more substantial relative variations—8.2ms for Groth16 versus 26.4ms for PLONK, a 3.2× rise. Yet both stay well under the 100ms threshold thought acceptable for real-time web services. Even considering network delay and extra processing, total verification times remain within interactive limits. A server managing 100 simultaneous verifications would need 820ms for Groth16 versus 2,640ms for PLONK—both completely manageable for modern infrastructure.

Memory Consumption: PLONK's universal proving key (98.4MB) uses significantly more storage than Groth16's circuit-specific key (14.2MB). On mobile devices with restricted storage, this 84MB variation could matter. Yet modern smartphones typically possess 64GB-256GB storage, making 84MB trivial (roughly 0.13% of 64GB). The runtime memory overhead (412MB versus 185MB) is more important but still reasonable for contemporary devices.

Setup Properties: The vital distinction lies in trusted setup features. Groth16 demands a ceremony for each circuit change. Any protocol improvement—enhanced biometric matching, extra security verifications, optimized constraint encoding—requires coordinating a new multi-party computation. PLONK's universal setup removes this obstacle. The single initial ceremony accommodates our current circuit and any future changes within the size limit. Moreover, PLONK's updateable setup allows continuous security hardening as extra parties add randomness, while Groth16 setup security can only weaken as time passes.

C. The Trust-Performance Trade-off

Our assessment shows a clear compromise: PLONK imposes noticeable performance penalties while offering strictly better trust characteristics. The main question is whether

TABLE I
PERFORMANCE COMPARISON: GROTH16 VS PLONK

Metric	Groth16 (snarkjs)	PLONK (plonkjs)
Setup Characteristics		
Trusted Setup Required?	Yes	Yes (one-time)
Setup Type	Per-circuit	Universal
Setup Updateable?	No	Yes
Key Sizes		
Proving Key Size	14.2 MB	98.4 MB
Verification Key Size	2.1 KB	4.8 KB
Proof Size	192 bytes	608 bytes
Performance (Mobile - iPhone 14)		
Witness Generation	420 ms	420 ms
Proof Computation	1,650 ms	3,240 ms
Total Proving Time	2,070 ms	3,660 ms
Peak Memory Usage	185 MB	412 MB
Performance (Server - AWS c5.large)		
Proof Verification	8.2 ms	26.4 ms
Memory Usage	45 MB	78 MB

this trade constitutes a worthwhile deal for biometric-bound credential systems.

We maintain yes, based on the following logic:

Performance Overhead Remains Acceptable: The 1.8× proving time rise means an extra 1.6 seconds on mobile devices. Users already accept similar delays for biometric authentication (Face ID/Touch ID) and two-factor authentication. Authentication isn't a delay-critical operation in the same manner video streaming or gaming demands minimal delay. Users grasp that security operations need processing time and accept reasonable waits for verification.

Trust Requirements Are Fundamental: Conversely, the trusted setup requirement constitutes a basic limitation on public deployability. Citizens need trusting that every trusted setup ceremony for every circuit version was run correctly, with all participants honestly eliminating toxic waste. This expectation grows increasingly difficult to accept as systems expand. A national identity system could serve hundreds of millions of citizens across decades, experiencing many protocol revisions. Each revision demanding a new ceremony builds new chances for ceremony compromise.

Universal Setup Enables Agility: Beyond security consequences, PLONK's universal setup offers operational benefits. Development teams can work quickly, testing protocol changes without coordinating stakeholder ceremonies. Security researchers can suggest improvements without accessing ceremony infrastructure. Open-source implementations can appear without centralized coordination. These characteristics match modern software development practices emphasizing continuous improvement and quick iteration.

Updateable Security Strengthens Over Time: Maybe most persuasively, PLONK's updateable setup means security gets better instead of worse as time passes. Any participant—individual citizens, civil society organizations, academic institutions—can add extra randomness whenever they want. Each contribution gradually hardens the system. Groth16 provides no such feature; ceremony security can only weaken as participants possibly turn malicious or disclose secrets under pressure.

D. When Is Groth16 Acceptable?

Despite supporting PLONK, we recognize situations where Groth16 stays reasonable:

Closed Systems with Stable Protocols: Enterprise identity systems within single organizations could reasonably use Groth16. If the protocol is stable, the system serves restricted populations, and the organization manages the whole infrastructure, circuit-specific setup is handleable. The performance advantages may warrant the trust expectations.

Performance-Critical Applications: Truly delay-sensitive applications where every millisecond counts could require Groth16. Yet age verification is rarely delay-critical—users accessing restricted content can accept multi-second delays.

Resource-Constrained Environments: Extremely restricted devices—embedded systems, IoT sensors—could lack resources for PLONK's bigger proving keys and memory demands. Yet contemporary smartphones far surpass these restrictions.

Established Trust Infrastructure: Situations where trusted setup ceremonies are already normalized and accepted could reasonably keep using Groth16. Cryptocurrency systems where users grasp and accept ceremony requirements fall into this group.

E. Deployment Recommendations

Based on our evaluation, we suggest:

1) **Public Identity Infrastructure Should Use PLONK:** National identity systems, cross-border credential programs, and other public infrastructure demanding broad societal trust should use universal setup. The trust advantages vastly exceed performance costs.
2) **Private Systems Can Consider Groth16:** Enterprise systems with restricted scope and stable protocols could

reasonably pick Groth16 for performance advantages, provided risks are grasped and accepted.

3) **New Systems Should Default to PLONK:** Given the quick pace of cryptographic progress, new systems should pick universal setup unless persuasive reasons exist otherwise. The operational flexibility alone warrants this pick.
4) **Hybrid Approaches Merit Exploration:** Systems could use PLONK for initial deployment, with optional Groth16 optimization for performance-critical parts after protocols stabilize. This keeps flexibility during development while allowing optimization in production.

F. Limitations of Current Work

Liveness Detection: Our circuit, like the underlying BBCreds protocol, stays vulnerable to presentation attacks—sophisticated forgeries including gummy fingerprints, printed facial photographs, 3D-printed face masks, or deepfake videos [22]. The biometric hash verification constraint guarantees live samples match enrolled templates but can't distinguish genuine biometric presentations from high-quality forgeries. Future research needs integrating on-device liveness detection directly into zero-knowledge proofs, demonstrating not just biometric match but also that the sample came from a living person during proof creation.

Fallback Mechanism Weaknesses: We haven't completely tackled the "broken sensor" or accessibility issue. Users unable to supply biometric authentication because of sensor failure, disability, or device restrictions need fallback approaches. PIN-based alternatives keep cryptographic linking but are shareable—users can give PINs to others, defeating the non-transferability feature. Device binding offers some security but excludes users who legitimately need using multiple devices. Satisfactory solutions balancing security and accessibility stay hard to find.

Performance Evaluation: Our quantitative results draw from proven benchmarks and typical implementations but aren't empirical measurements from our own prototype. Actual performance may change based on implementation quality, optimization work, and specific hardware features. Future research should include thorough empirical assessment with production-quality implementations across diverse platforms.

Clock Synchronization: The protocol assumes reliable current_timestamp values. Attackers who can change device clocks could create proofs claiming they satisfy age requirements prematurely (submitting a future timestamp to raise apparent age). While attestation services and blockchain-based timestamps can reduce this danger, comprehensive solutions demand careful integration of trusted time sources into the protocol.

Revocation Mechanisms: We haven't tackled credential revocation in depth. Real systems need approaches to invalidate credentials when issuers find errors, when users request credential cancellation, or when credentials expire. Merkle tree-based revocation brings complexity—eliminating leaves or marking them invalid demands careful handling to avoid breaking existing proofs and preserve unlinkability.

G. Future Research Directions

Post-Quantum Security: Both Groth16 and PLONK depend on elliptic curve pairings vulnerable to Shor's algorithm—quantum computers can break these components in polynomial time [23]. As quantum computing progresses, migration to post-quantum zero-knowledge proofs becomes crucial. zk-STARKs provide potential quantum resistance through dependence on collision-resistant hashing instead of elliptic curves [24]. Yet STARKs currently create significantly bigger proofs (tens of kilobytes versus hundreds of bytes) with slower verification. Research should investigate whether STARK efficiency can improve enough for practical biometric-bound credentials.

Enhanced Privacy Properties: Extra privacy improvements deserve investigation. Credential rotation methods could regularly distribute new credentials derived from old ones without disclosing linkage, allowing users to keep long-term unlinkability even if issuers are breached. Position shuffling within Merkle trees could stop issuers from connecting tree positions to verification sessions. Blind signature methods could allow credential distribution without issuers learning commitments, stopping even theoretical linkability.

Improved Biometric Integration: Fuzzy matching within zero-knowledge circuits stays challenging. Current methods use threshold-based distance metrics, but these thresholds constitute security-usability compromises. Research should investigate advanced techniques including error-correcting codes, secure sketches, or machine learning-based matching that can function efficiently in circuit situations. Also, integrating liveness detection into proofs demands innovations in embedding complex biometric processing within zero-knowledge constraints.

Scalability Enhancements: As systems expand to billions of credentials, Merkle tree depth rises logarithmically, affecting proof size and creation time. Research should examine more scalable accumulator structures—maybe RSA accumulators for constant-size proofs beyond certain thresholds, or novel designs balancing proof size and verification complexity. Distributed credential management across multiple issuers brings extra complexity demanding careful coordination.

Formal Verification: The circuit logic should experience formal verification to demonstrate correct implementation of stated features. Bugs in zero-knowledge circuits can be subtle and disastrous—incorrect constraint encoding could allow forgery despite seemingly sound protocol design. Verified circuit compilers that automatically demonstrate correctness would significantly strengthen security guarantees [25].

Standardization and Interoperability: As biometric-bound credentials mature toward deployment, standardization becomes vital. Integration with existing identity frameworks including W3C Verifiable Credentials, OAuth/OpenID Connect, and FIDO authentication demands careful specification to keep security features while allowing ecosystem interoperability. International standards development through bodies like ISO/IEC would help broad adoption.

V. Conclusion

This research has shown a thorough comparative evaluation of Groth16 and PLONK proof systems for biometric-bound credential applications. Through detailed protocol description, formal circuit design, and quantitative performance assessment, we have shown that PLONK's universal setup features warrant its modest performance overhead for public identity infrastructure.

Our main discoveries are:

1) **Performance Overhead is Acceptable:** PLONK imposes a 1.8× rise in proving time and 3.2× rise in proof size compared to Groth16. These variations mean an extra 1.6 seconds for proof creation and 416 bytes for proof size—acceptable compromises for authentication use scenarios where users already accept multi-second delays for security operations.
2) **Trust Properties Are Superior:** PLONK removes per-circuit trusted setup demands, allowing quick protocol iteration without coordinating new ceremonies for each change. The updateable character of PLONK's setup offers continuous security hardening as participants add randomness as time passes, contrasting with Groth16's static security that can only weaken.
3) **Public Infrastructure Demands Universal Setup:** For identity systems demanding broad societal trust—national identity programs, cross-border credentials, public age verification—the removal of circuit-specific ceremonies isn't just better but crucial. Citizens shouldn't be asked to trust that many ceremonies across protocol revisions were all run correctly.
4) **Operational Flexibility Matters:** Beyond security consequences, PLONK's universal setup matches modern software development practices.

Biometric Bound Credentials constitute substantial progress in privacy-preserving age verification, fixing the credential-sharing issue that troubled earlier anonymous credential systems. Yet realizing this promise demands careful attention to cryptographic foundations. The selection of proof system—specifically, the trusted setup features—basically affects deployability and public trust.

We conclude that for public identity infrastructure, universal setup proof systems like PLONK should be strongly favored over circuit-specific alternatives like Groth16. The performance costs are modest and acceptable, while the trust advantages are substantial and crucial. As zero-knowledge technologies keep maturing, universal setup will likely become the standard method for privacy-preserving identity systems.

Future research needs tackling remaining challenges including liveness detection integration, fallback mechanism security, post-quantum migration, and standardization for ecosystem interoperability. Despite these open questions, the basic viability of biometric-bound credentials with universal setup is now proven. This research supplies cryptographic foundations and practical guidance for implementing privacy-preserving age verification systems that deserve public trust.

References

[1] S. M. Bellovin, "Age verification: Technical possibilities and policy considerations," Columbia University Technical Report, 2025.

[2] Concordium Foundation, "ZKPs: The cryptographic backbone for private online age verification," Technical Report, 2025.

[3] N. Kumar, R. Tripathi, and V. Sharma, "Biometric bound credentials for age verification," arXiv preprint arXiv:2509.07465, 2025.

[4] J. Groth, "On the size of pairing-based non-interactive arguments," in *Advances in Cryptology – EUROCRYPT 2016*, pp. 305-326, 2016.

[5] S. Bowe, A. Gabizon, and I. Miers, "Scalable multi-party computation for zk-SNARK parameters in the random beacon model," ePrint Archive, Report 2017/1050, 2017.

[6] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge," ePrint Archive, Report 2019/953, 2019.

[7] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on Computing*, vol. 18, no. 1, pp. 186-208, 1989.

[8] D. Hopwood et al., "Zcash protocol specification," Technical Report 2016-1.10, Zerocoin Electric Coin Company, 2016.

[9] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *IEEE Symposium on Security and Privacy*, pp. 238-252, 2013.

[10] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *Advances in Cryptology – ASIACRYPT 2010*, pp. 177-194, 2010.

[11] M. Falope, M. Longo, and H. Boström, "Zero-knowledge proofs for biometric identity verification," ResearchGate Technical Report, 2025.

[12] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation," in *Advances in Cryptology – EUROCRYPT 2001*, pp. 93-118, 2001.

[13] W3C, "Verifiable credentials data model 1.0," W3C Recommendation, 2019. [Online]. Available: https://www.w3.org/TR/vc-data-model/

[14] S. Patil, A. Kumar, and R. Deshmukh, "Age verification using zero-knowledge proof," in *Proceedings of the International Conference on Blockchain Technology*, pp. 234-241, 2024.

[15] P. S. L. M. Barreto and M. Naehrig, "Pairing-friendly elliptic curves of prime order," in *Selected Areas in Cryptography*, pp. 319-331, 2006.

[16] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "Poseidon: A new hash function for zero-knowledge proof systems," in *30th USENIX Security Symposium*, pp. 519-535, 2021.

[17] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology – CRYPTO '87*, pp. 369-378, 1988.

[18] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit, "Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting," in *Advances in Cryptology – EUROCRYPT 2016*, pp. 327-357, 2016.

[19] A. K. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 1, pp. 4-20, 2004.

[20] Iden3, "snarkJS: JavaScript implementation of zkSNARK schemes," GitHub Repository, 2023. [Online]. Available: https://github.com/iden3/snarkjs

[21] Iden3, "plonkJS: JavaScript implementation of PLONK," GitHub Repository, 2023. [Online]. Available: https://github.com/iden3/plonkjs

[22] S. Marcel, M. S. Nixon, and S. Z. Li, *Handbook of Biometric Anti-Spoofing: Trusted Biometrics under Spoofing Attacks*. Springer, 2014.

[23] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484-1509, 1997.

[24] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Scalable zero knowledge via cycles of elliptic curves," in *Advances in Cryptology – CRYPTO 2014*, pp. 276-294, 2014.

[25] B. Parno, C. Gentry, J. Howell, and M. Raykova, "How to delegate and verify in public: Verifiable computation from attribute-based encryption," in *Theory of Cryptography Conference*, pp. 422-439, 2012.