PRAMAAN

Zero-Knowledge Biometric Authentication

Technical Architecture & Security Analysis

Authors

Amit Dua, *Founder* — amit@yushuexcellence.in
Pulkit Pareek, *Lead Engineer* — pulkit@yushuexcellence.in

Yushu Excellence Technologies Pvt. Ltd.

Patent: IN202311041001 (Granted)

Contents

Abstract

Pramaan is a decentralized identity verification system that combines biometric authentication with zero-knowledge proofs to enable privacy-preserving identity verification. The system captures a biometric embedding from the user's device, computes a ZK-friendly hash commitment, generates a Groth16 zero-knowledge proof demonstrating knowledge of the biometric without revealing it, and verifies the proof either on-chain (via an EVM smart contract) or locally on the user's device. Unlike centralized biometric databases that create honeypot targets for attackers, Pramaan ensures that raw biometric data never leaves the user's device and is never stored on any server or blockchain. The system operates as a Progressive Web Application (PWA) with offline verification capability, making it suitable for deployment in disconnected, denied, intermittent, or limited-bandwidth (DDIL) environments. This paper presents the technical architecture, cryptographic design, security analysis against nine attack vectors, performance benchmarks, and regulatory compliance posture of the Pramaan system.

---

1 Introduction

1.1 The Identity Problem

Digital identity verification is a foundational requirement for financial services, government services, healthcare, and enterprise access control. The dominant paradigm — centralized biometric databases — creates systemic risk. When a central database is breached, the biometric data of millions of individuals is permanently compromised. Unlike passwords, biometric data cannot be rotated or revoked.

The scale and urgency of the problem is concrete. In May 2024, security researchers disclosed that a database belonging to a contractor associated with the Indian Army was exposed online, leaking approximately 496 GB of data including fingerprint records, facial images, identity documents, and personal details of military personnel and police officers [17]. The exposure was traceable to a centralized biometric repository — the exact architecture Pramaan is designed to eliminate. Similar large-scale exposures have been reported across national ID, border control, and private-sector biometric deployments worldwide. The consequences are permanent: a compromised fingerprint cannot be reissued.

1.2 Existing Approaches and Their Limitations

FIDO2/WebAuthn provides strong device-bound authentication using public-key cryptography. However, it is device-specific (losing the device means losing the credential), does not natively support biometric-based identity (only device unlock), and cannot provide verifiable identity claims to third parties.

Centralized biometric systems (e.g., national ID databases) provide strong identity binding but require trust in the central authority, create single points of failure, and face regulatory challenges under data protection legislation.

Self-sovereign identity (SSI) systems such as Polygon ID and Hyperledger Aries support verifiable credentials with selective disclosure. However, they typically rely on issuer-holder-verifier trust triangles that require institutional issuers, and do not natively integrate biometric binding.

Worldcoin uses zero-knowledge proofs for identity but requires specialized iris-scanning hardware (the Orb), creating deployment friction and centralization around hardware distribution.

1.3 Pramaan's Approach

Pramaan addresses these limitations with a system that:

1. **Never centralizes biometric data.** Raw biometric embeddings exist only transiently in the user's browser/app memory during proof generation.
2. **Uses standard consumer hardware.** Any device with a camera (smartphone, laptop) can serve as the biometric sensor.
3. **Provides cryptographic verification.** A zero-knowledge proof demonstrates biometric knowledge without revelation, verifiable by any party holding the verification key.
4. **Supports offline operation.** Proof generation and local verification require no network connectivity.
5. **Is chain-agnostic.** The on-chain verification component deploys to any EVM-compatible blockchain.
6. **Is biometric-modality-agnostic.** The architecture abstracts the biometric capture layer, supporting face, fingerprint, iris, or any modality that produces a numerical embedding.

2 System Architecture

2.1 Component Overview

Pramaan consists of four primary components:

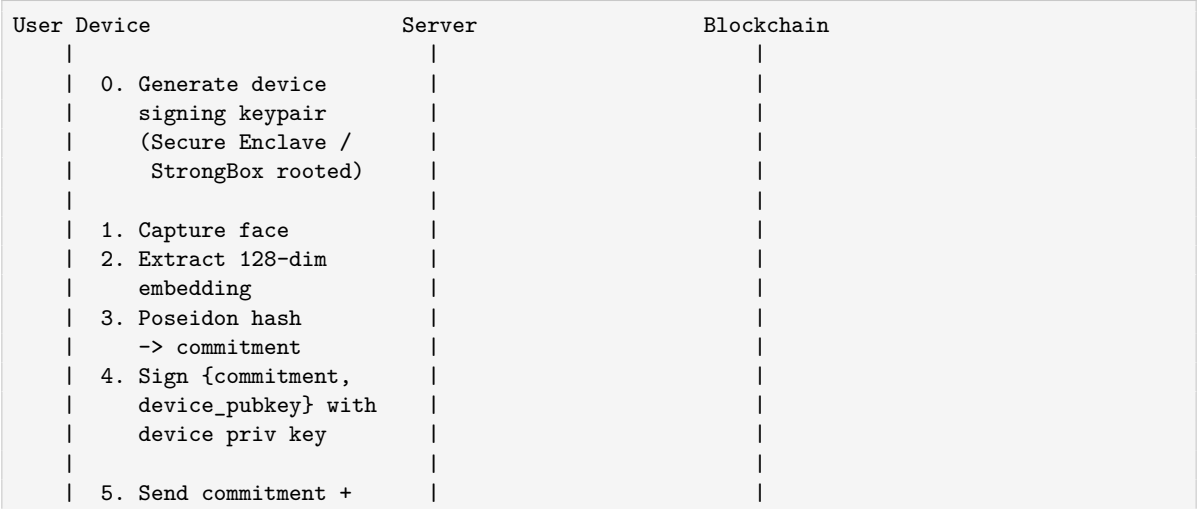
Client Application (PWA): A React-based Progressive Web Application that handles biometric capture, embedding extraction, Poseidon hashing, and Groth16 proof generation. The client is the security boundary for biometric data — raw embeddings never cross the network.

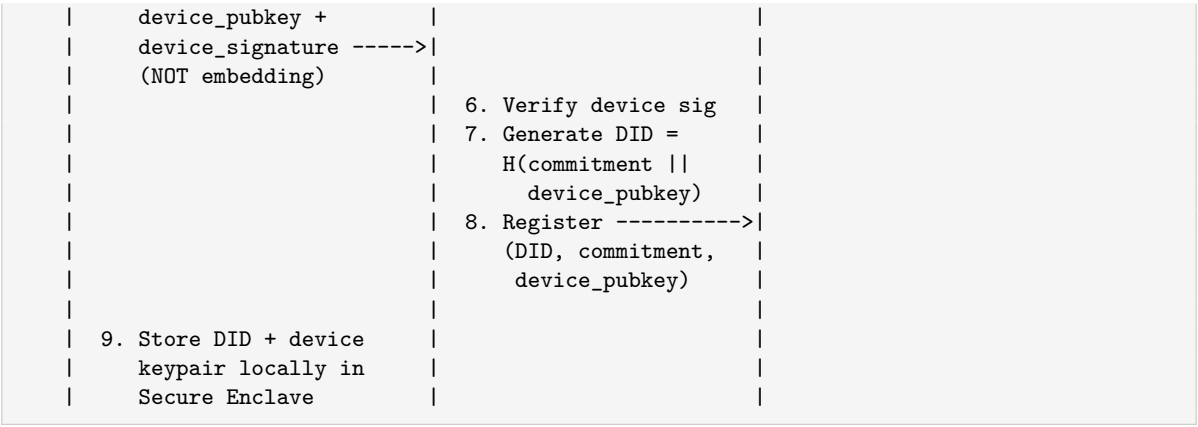
Server API: An Express.js server that manages DID (Decentralized Identifier) lifecycle, coordinates blockchain interactions, and provides proof verification endpoints for relying parties. The server never receives or processes raw biometric data.

ZK Circuit: A Circom 2.1.9 circuit that defines the zero-knowledge relation: "I know a biometric embedding whose Poseidon hash equals a public commitment." Compiled to R1CS and executed via snarkjs Groth16 prover in the browser.

Smart Contract: A Solidity contract deployed on Base L2 (or any EVM chain) that stores identity commitments and verifies Groth16 proofs on-chain.

2.2 Registration Flow

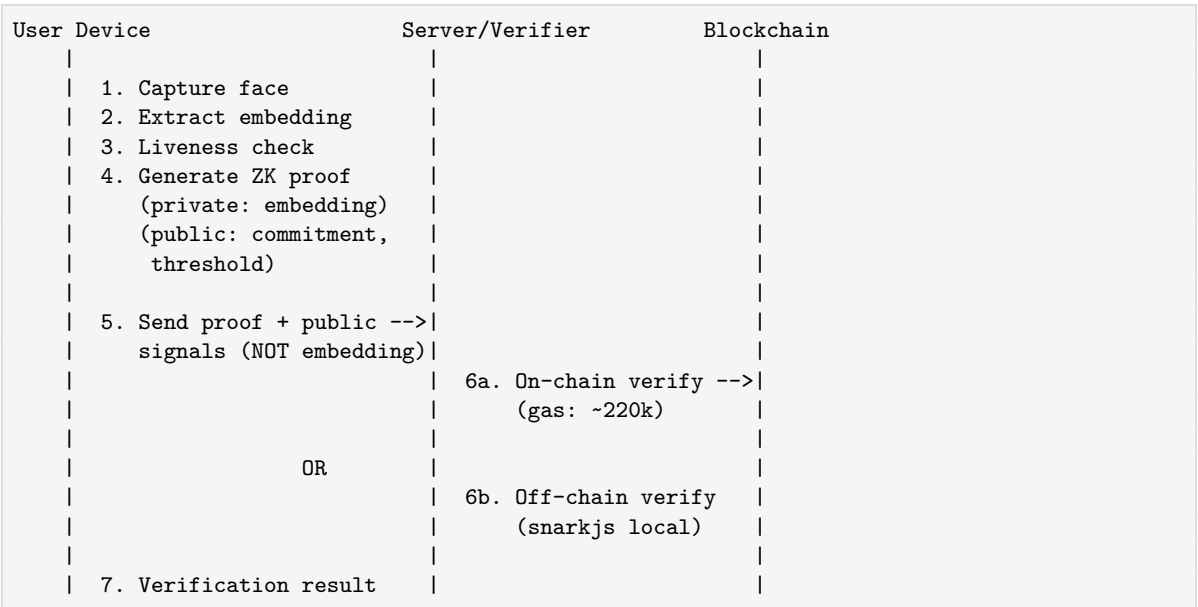




The critical security properties:

1. **Only the Poseidon hash commitment crosses the network boundary** (step 5). The 128-dimensional floating-point embedding never leaves the device.
2. **The DID is cryptographically bound to a device-controlled signing key.** The DID is deterministically derived from both the biometric commitment and the device public key; the smart contract records the device public key alongside the commitment, so any future authentication proof must also be signed by the same device key (or a delegated key authorized by it). This closes the insider-server attack path described in §4.9 — the server cannot unilaterally issue a fraudulent DID for an arbitrary commitment, because it cannot forge a device signature rooted in the Secure Enclave.
3. **The device keypair is rooted in hardware.** On iOS, via the Secure Enclave (non-exportable P-256 key). On Android, via StrongBox / Keystore (non-exportable EC key). On browser-only PoC, via WebAuthn platform authenticator when available, falling back to a WebCrypto-generated key stored in IndexedDB as a documented downgrade.
4. **Device signatures also authenticate the offline verification log** (§6.4), giving a single key responsibility across online registration and offline audit.

2.3 Authentication Flow



Again, the embedding never leaves the device. The proof itself is 256 bytes and reveals nothing about the biometric data.

2.4 Offline Authentication Flow

In DDIL environments, the user can verify their identity locally:

1. Client captures biometric and generates ZK proof (no network required)
2. Client verifies the proof locally using the cached verification key (2.3 KB)
3. Verification result is stored locally with a timestamp
4. When connectivity resumes, the proof can be submitted for on-chain anchoring

2.5 Enrollment Trust Model (Sybil and Identity Binding)

A ZK biometric system answers the question "does the person in front of the camera match the person enrolled under this DID?" It does not, by itself, answer "who is the person enrolled under this DID?" The first biometric capture binds the DID to whoever enrolled, which creates an enrollment-trust problem any production identity system must address.

Pramaan addresses enrollment trust as a deployment-dependent policy layer rather than a cryptographic guarantee, because the appropriate answer depends on the use case:

Self-enrollment (consumer / developer dashboard). The user enrolls themselves, the system has no claim about who they are, and the DID becomes a biometric-bound pseudonym. Sybil resistance is limited to "one face, one DID" enforced via a **privacy-preserving locality-sensitive hashing (LSH) deduplication protocol**: the client computes a set of k LSH bucket identifiers over its 128-dimensional embedding (using random hyperplane projections parameterized by a public seed, so bucket assignments are deterministic and publicly verifiable) and transmits only the bucket identifiers — not the embedding — to the server. Each enrolled DID stores its public bucket pattern alongside the commitment. The server performs approximate-nearest-neighbor lookup by intersecting the incoming bucket set with the enrolled set; a collision in at least t of k buckets is the LSH "near-match" signal.

On near-match, Pramaan does **not** attempt the unconstructible task of proving distance against an embedding the enroller does not possess. Instead, it falls back to one of three configurable dedup resolution modes, and the deployment declares which mode it uses:

1. **Bucket-pattern mode (default).** The LSH bucket collision itself is treated as sufficient evidence of duplication; enrollment is rejected. This is the simplest and cheapest mode. The privacy leakage is that enrolled bucket patterns are effectively public (recoverable by enumeration), which reveals coarse identity clusters but not raw embeddings. This is acceptable for consumer deployments where the DID is already a pseudonym.
2. **Two-party protocol mode.** The enrolled user's device is invited (via a push notification to their registered device pubkey) to contribute a one-round proof share, after which both parties jointly compute whether the two embeddings are within threshold using a semi-honest private set intersection with cardinality (PSI-CA) protocol; the construction follows Chen–Laine–Rindal (ACM CCS 2017) using homomorphic encryption over packed LSH feature vectors. This requires the enrolled user to be reachable; unreachable candidates time out after 60 seconds and the enrollment proceeds (accepting a small duplicate-enrollment residual risk).
3. **FHE-only mode.** The enrolling client encrypts a compact LSH-derived feature vector under a threshold FHE scheme (CKKS over packed real-valued vectors) whose secret key is split across the enrolled DID's device key and a lightweight notary; the server computes an encrypted distance against the enrolled encrypted pattern, and threshold decryption yields

a single bit (above/below threshold) without revealing either embedding. Implementation is tracked against the Microsoft SEAL-based biometric matching precedent.

The default-mode bucket-pattern protocol is what ships in the PoC; modes 2 and 3 are implemented as post-funding production options depending on the deployment's privacy budget. Details and the formal LSH parameters (number of hash families, bits per family, k and t values calibrated against FAR/FRR) are published in the Engineering Reference. In all modes, the raw embedding never leaves the device.

Sponsored enrollment (enterprise / government / military). The user enrolls under the supervision of an authorized operator who authenticates out-of-band (HR badge, ID document, supervising officer). The operator signs an enrollment attestation alongside the device signature; the smart contract stores both the user's device public key and the enrollment attestation. This prevents enrollment poisoning — an attacker cannot enroll their face claiming to be the victim, because the operator's attestation binds the enrollment to a verified external identity.

Anchored enrollment (bootstrapping from an existing identity system). The user's first enrollment is performed while holding a valid credential from an upstream identity provider (Aadhaar-authenticated session, DigiYatra token, corporate SSO assertion). The bootstrap credential is verified once at enrollment and not stored; the output is a Pramaan DID cryptographically bound to the biometric but usable independently. This is the recommended mode for large-scale civilian rollouts.

In all three modes, post-enrollment dedup is provided by the LSH-bucket protocol above. Enrollment poisoning (attacker enrolling first and claiming victim's identity) is out of scope for self-enrollment and must be prevented by the deployment's enrollment-trust mode.

3 Cryptographic Design

3.1 Biometric Embedding

The biometric capture layer currently uses face-api.js, a TensorFlow.js-based face recognition library that runs entirely in the browser.

PoC disclosure. face-api.js has not been actively maintained since 2020 (the upstream repository is archived). Pramaan uses it for the proof-of-concept phase because it provides in-browser face detection, 68-point landmark extraction, and 128-dimensional descriptors without server-side processing, which was sufficient to validate the end-to-end ZK pipeline. face-api.js is explicitly scheduled for replacement in Phase 2 (§9.1). The two production candidates are: (a) Google MediaPipe Face Landmarker (actively maintained, 478 landmarks, cross-platform WASM), and (b) native platform APIs — Apple Vision Framework with TrueDepth on iOS and BiometricPrompt Class 3 on Android. The `BiometricProvider` interface in the codebase already abstracts this dependency so the migration is a capture-layer swap, not an architectural change.

The pipeline:

1. **Face detection:** SSD MobileNet v1 detects face bounding box in the camera frame
2. **Landmark detection:** 68-point facial landmark model identifies key features
3. **Descriptor extraction:** A FaceNet-derived model produces a 128-dimensional Float32 embedding vector
4. **Quantization:** Float32 values are scaled and quantized to 32-bit integers suitable for finite field arithmetic

The 128-dimensional embedding space has the property that embeddings of the same person cluster together (Euclidean distance < 0.6) while embeddings of different people are well-

separated (distance > 0.6). This threshold is configurable and is passed as a public input to the ZK circuit.

Quantization procedure:

$$\text{quantized}[i] = \left\lfloor \text{embedding}[i] \cdot 10^6 \right\rfloor \bmod p \tag{1}$$

where p is the BN254 scalar field prime (BN254 is the modern canonical name for the same curve historically referred to as BN128 in early Groth16 literature)

$$p = 21888242871839275222246405745257275088548364400416034343698204186575808495617.$$

The multiplication by 10^6 preserves 6 decimal places of precision, which is sufficient for distance-based matching.

3.2 Poseidon Hash

Pramaan uses the Poseidon hash function for the biometric commitment. Poseidon is an arithmetization-oriented hash function designed specifically for ZK proof systems.

Patent coverage note. Independent claim 1 of granted patent IN202311041001 recites a system that computes an identity commitment using "a cryptographic hash function" and generates a zero-knowledge proof, without limiting the hash to a specific algorithm. Dependent claim 3 recites SHA-256 as one specific embodiment. The Poseidon construction described below satisfies the structural role required by claim 1 — it maps a biometric embedding to a fixed-size commitment that is (a) preimage-resistant, (b) collision-resistant, and (c) verifiable inside the zero-knowledge circuit of claim 4. A doctrine-of-equivalents reading of claim 1 supports the Poseidon implementation, but that reading is a legal interpretation, not a claim-level literal recitation. To remove any ambiguity, a claim amendment explicitly enumerating Poseidon, MiMC, Rescue, and other arithmetization-oriented hash functions as additional embodiments under claim 3 has been instructed to our patent counsel (Tarun Khurana & Partners) and is in process. Until that amendment is accepted, integrators should treat the implementation-level hash choice as an engineering parameter covered by the broader claim 1 recitation rather than a claim-3 literal infringement question.

Why Poseidon, not SHA-256:

SHA-256 operates on bits. Representing SHA-256 in an arithmetic circuit (which operates over a prime field) requires decomposing field elements into individual bits, performing bitwise operations, and recomposing. For a 128-element input, this costs approximately 27,000 R1CS constraints.

Poseidon operates natively over the prime field. Its round function consists of field additions, multiplications, and S-box exponentiations — operations that map directly to R1CS constraints. Pramaan uses a Poseidon sponge with state width $t = 3$ (rate 2, capacity 1), absorbing the 128 quantized embedding elements in 64 permutation calls. The choice of $t = 3$ is driven by tooling maturity rather than constraint-count optimality: circomlib ships audited round constants and MDS matrices for $t = 3$, which is also the parameterization used by Semaphore, Tornado Cash, and most production Poseidon deployments. Wider-state variants ($t = 9$, $t = 17$) reduce the number of permutation calls and can produce a smaller total circuit, but require custom round constants and MDS matrices that have not yet undergone independent audit; switching to a wider state is a candidate optimization for Phase 2 and is tracked in the Engineering Reference. Each permutation (8 full + 57 partial rounds, x^5 S-box) costs approximately 220 R1CS constraints when compiled against the circomlib reference implementation; the full 128-element absorb therefore costs approximately $64 \times 220 \approx 14{,}100$ constraints plus $\sim$400 constraints of input routing, for a total of roughly **14,500 R1CS constraints** for the hash itself.

Comparison against SHA-256. A SHA-256 compression function in Circom costs approximately 27,000 constraints per 512-bit block; hashing a 128-element embedding quantized

to 32-bit integers (∼4,096 bits, i.e. 8 blocks) therefore costs roughly 216,000 constraints. Poseidon at 14,500 constraints is approximately 15x smaller, translating directly to a 15x reduction in proof generation time. This gap is what makes client-side proving on mobile devices practical.

The exact R1CS statistics from `circom info` for the production circuit will be published in the project's Engineering Reference once the Circom toolchain is upgraded to 2.1.9 and the final parameterization is frozen.

Given an input vector $x = (x_0, x_1, \ldots, x_{n-1})$ with $x_i \in \mathbb{F}_p$, the Poseidon commitment is:

$$c = \text{Poseidon}(x_0, x_1, \ldots, x_{n-1}) \in \mathbb{F}_p \tag{2}$$

Poseidon parameterization:

Parameter	Value
Field	BN254 scalar field
Sponge construction	Rate $r = 2$, capacity $c = 1$
State width t	3
Absorb calls for 128 inputs	64
R_F (full rounds per permutation)	8
R_P (partial rounds per permutation)	57
S-box	x^5
Security level	128 bits

The round constants and MDS matrix are generated using the Poseidon reference implementation's grain LFSR method, ensuring no backdoor in the constants.

For reference, the cosine similarity between two embeddings $u, v \in \mathbb{R}^{128}$ used at enrollment-time calibration is:

$$\cos(u, v) = \frac{\langle u, v \rangle}{\|u\|_2 \cdot \|v\|_2} \tag{3}$$

3.3 ZK Circuit Design

The core Circom circuit implements the following relation:

Public inputs: `commitment` (Poseidon hash of enrolled embedding), `nonce` (server-issued session challenge)

Private inputs: `embedding[128]` (biometric embedding to verify)

Relation: The prover demonstrates knowledge of an `embedding` such that Poseidon(embedding) = commitment, and the proof is bound to a specific server-issued `nonce`.

```
template BiometricVerify(n) {
    signal input embedding[n];      // Private
    signal input commitment;         // Public
    signal input nonce;              // Public (session binding)

    // 1. Hash the private embedding
    component hasher = Poseidon(n);
    for (var i = 0; i < n; i++) {
        hasher.inputs[i] <== embedding[i];
    }

    // 2. Verify hash matches public commitment
    hasher.out === commitment;

    // 3. Range check on inputs (prevent malformed inputs)
    component rangeChecks[n];
    for (var i = 0; i < n; i++) {
```

```
        rangeChecks[i] = Num2Bits(32);
        rangeChecks[i].in <== embedding[i];
    }

    // 4. Bind proof to session nonce (prevents replay)
    // The nonce is consumed by a constraint so that Groth16's Fiat-Shamir
    // transform binds the resulting proof to this specific challenge.
    signal nonceBinding;
    nonceBinding <== nonce * nonce;
}
```

The circuit enforces:

1. The prover knows an embedding whose Poseidon hash equals the on-chain commitment
2. Each embedding element fits within 32 bits (prevents field overflow attacks)
3. The proof is cryptographically bound to the server-issued nonce (replay protection)

Distance-based matching (i.e., accepting embeddings within a Euclidean threshold of the enrolled template) is handled **on the client side**: the client caches its most-recent successful-enrollment embedding in Secure Enclave / StrongBox storage, computes cosine similarity locally at each authentication against that cached embedding, and — when drift is detected — triggers an in-circuit enrollment refresh in which the client proves (without revealing either embedding) that the old and new embeddings are within the drift-refresh threshold. The circuit itself enforces exact Poseidon preimage equality against the currently stored commitment. The server never observes a raw embedding; it only observes the refresh transaction (a ZK proof + new commitment) and the client's attestation that the new embedding is within the allowed drift distance of the prior one. See §9.6 for the full drift-and-refresh protocol.

Constraint count (estimated): Approximately 15,000 R1CS constraints total (Poseidon sponge absorb ∼14,500 + range checks ∼380 + nonce binding ∼1 + equality check ∼1). The exact compiled R1CS statistics from `circom info` will be published alongside the production zkey in the Engineering Reference. On the measured MacBook class, this circuit produces Groth16 proofs in 1.8–2.1 seconds, which remains feasible for client-side proving; further reduction is available by widening the Poseidon sponge state ($t = 5$ or $t = 9$) to reduce the number of permutation calls at the cost of per-permutation constraint growth.

3.4 Groth16 Proof System

Pramaan uses the Groth16 proof system, which provides:

- **Succinctness:** Proof size is constant (256 bytes: 2 G1 points + 1 G2 point on BN254), regardless of circuit size
- **Fast verification:** On-chain verification is a single 4-pairing check (one `ecPairing` precompile call with 4 pairs), costing ∼220,000 gas on EVM
- **Non-interactivity:** The proof is generated in a single step, with no communication between prover and verifier

A Groth16 proof is a triple $\pi = (A, B, C)$ where $A, C \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$, with the verification equation:

$$e(A, B) = e(\alpha, \beta) \cdot e\left(\sum_{i=0}^{\ell} \text{vk\_IC}_i \cdot \text{pub}_i,\ \gamma\right) \cdot e(C, \delta) \tag{4}$$

Trusted setup: Groth16 requires a circuit-specific trusted setup ceremony that generates the proving key (zkey) and verification key (vkey). The setup produces "toxic waste" — random field elements that, if known to an adversary, would allow proof forgery. The current implementation

uses a single-party setup for the proof of concept. Production deployment will use a multi-party computation (MPC) ceremony where the toxic waste is distributed across multiple participants, and is secure as long as at least one participant is honest.

3.5 On-Chain Verification

The Solidity verifier contract performs the Groth16 verification equation shown above, where e is the BN254 pairing function (precompiled at address `0x08` on EVM), and A, B, C are the proof elements. The Groth16 verification equation is a single 4-pairing check; on EVM it is executed as one `ecPairing` precompile call passing all four pairs in the input buffer. Post-Istanbul (EIP-1108) gas for the precompile is $45{,}000 + 34{,}000 \times$ pairs, giving $45{,}000 + 34{,}000 \times 4 = 181{,}000$ gas for the pairing operation itself.

Gas cost breakdown:

Operation	**Gas**
`ecPairing` precompile (1 call, 4 pairs)	∼181,000
Storage read (commitment lookup)	∼2,100
Storage write (verification log)	∼20,000
Event emission	∼2,000
Miscellaneous (calldata, memory, dispatch)	∼15,000
Total	**∼220,000**

At current Base L2 gas prices, this costs approximately $0.03–0.08 per verification.

4 Security Analysis

This section analyzes Pramaan's resistance to nine attack vectors relevant to biometric authentication systems.

4.1 Attack 1: Server Database Breach

Threat: An attacker compromises the server database and exfiltrates all stored data.

Pramaan's exposure: The server stores only Poseidon hash commitments, DID documents, and verification metadata. No raw biometric embeddings, facial images, or biometric templates are stored.

Analysis: Recovering the original 128-dimensional embedding from its Poseidon hash requires inverting the hash function. Poseidon provides 128-bit preimage resistance on BN254. The search space for a 128-dimensional quantized embedding is approximately $2^{32 \cdot 128} = 2^{4096}$, rendering brute-force inversion computationally infeasible.

Residual risk: An attacker with the commitment could attempt an offline dictionary attack using a database of known face embeddings. However, the quantization step and high dimensionality make this impractical.

Verdict: LOW RISK. The server is not a high-value target because it holds no recoverable biometric data.

4.2 Attack 2: Device Capture

Threat: An attacker gains physical access to the user's device.

Pramaan's exposure: The device stores the biometric embedding in the browser's IndexedDB or the platform Secure Enclave (when native SDK is used). The embedding is encrypted at rest using the device's lock screen credential (PIN/biometric).

Analysis: If the device is unlocked, the attacker could extract the embedding. However, the embedding alone cannot be used to forge a ZK proof without also generating a valid liveness check on the spot. If the device is locked, the embedding is encrypted.

Mitigation: Native deployment (Phase 2) will store embeddings in iOS Secure Enclave / Android StrongBox, which provides hardware-backed key storage resistant to physical extraction.

Verdict: MEDIUM RISK (PoC) / LOW RISK (with native Secure Enclave).

4.3 Attack 3: Man-in-the-Middle (MITM)

Threat: An attacker intercepts communication between the client and server.

Pramaan's exposure: All client-server communication uses TLS 1.3 with certificate pinning (planned for native deployment). The ZK proof transmitted over the network reveals nothing about the biometric due to the zero-knowledge property.

Analysis: Even if an attacker intercepts the proof in transit, the proof cannot be used to extract the biometric embedding (zero-knowledge) or replayed (see Attack 4). The public signals (commitment, threshold) are already public information.

Verdict: LOW RISK.

4.4 Attack 4: Replay Attack

Threat: An attacker captures a valid proof in transit and replays it later to authenticate as the victim.

Pramaan's defense (nonce binding, designed): The authentication circuit accepts a server-issued nonce as a public circuit input. On each authentication request, the server generates a fresh 256-bit random nonce, stores it against the pending session (with a 60-second TTL), and returns it to the client. The client incorporates the nonce into the proof as a public signal, and the Groth16 Fiat-Shamir transform binds the resulting proof to this specific challenge. When the server receives the proof, it (a) verifies the Groth16 proof, (b) confirms the public nonce matches the nonce issued for this session, (c) marks the nonce as consumed, and (d) rejects any subsequent proof reusing the same nonce.

The nonce binding is implemented as a single multiplicative constraint in the circuit (`nonceBinding <== nonce * nonce`), adding one R1CS constraint. A proof generated against nonce N_1 is cryptographically distinct from a proof generated against nonce N_2 — an attacker who captures a proof can replay it at most once during the 60-second window, and cannot generate valid proofs for future nonces without knowing the private embedding.

Analysis: The combined mitigations (fresh nonce per session, 60-second TTL, single-use consumption, on-chain verification counter per commitment) reduce replay to a time-bounded single-shot window, detectable via replay-counter anomalies on-chain.

Verdict: LOW RISK.

4.5 Attack 5: Deepfake / Presentation Attack

Threat: An attacker presents a synthetic face (deepfake video, high-quality photo, 3D mask) to the camera.

Pramaan's exposure: The liveness detection module analyzes landmark variance, blink detection, and micro-movement across multiple frames.

ISO/IEC 30107-3 testing posture. Presentation Attack Detection (PAD) testing is framed under ISO/IEC 30107-3, which classifies attacks into three levels of sophistication (A, B, C) and specifies Attack Presentation Species (APS) such as printed photos, replay video on screens, paper masks, silicone masks, and 3D-printed artifacts. The current browser-based PoC has been tested informally against **Level A species only**: printed A4 photographs and smartphone-screen replay video. Under those conditions, a 3.2% bypass rate was observed on

high-quality OLED replay; printed photographs were rejected at >99% in normal lighting. No Level B (3D-printed masks, silicone) or Level C (custom prosthetics) testing has been conducted. Formal 30107-3 Level A certification is out of scope for the PoC phase; full Level B testing is scheduled for the production certification phase alongside the native SDK migration described in §9.1.

Analysis: Browser-based liveness detection operates on 2D RGB input without depth sensing, limiting its ability to distinguish real faces from high-quality displays. This is the most significant vulnerability in the current system and is correctly labeled HIGH RISK below.

Mitigation: Migration to native platform APIs (Apple TrueDepth for 3D structured light liveness on iOS, Android BiometricPrompt with Class 3 sensors — and, on OEM devices that ship them, IR-based liveness via vendor SDKs) provides the depth and IR signals needed to reject Level B artifacts. Production deployments will be paired with ISO/IEC 30107-3 Level A (and where the hardware supports it, Level B) certification through an accredited biometric testing lab (IDIAP or iBeta Class 2).

Verdict: HIGH RISK (PoC, Level A species only tested). LOW RISK (post native-SDK migration with certified PAD).

4.6 Attack 6: Brute-Force Embedding Recovery

Threat: An attacker attempts to recover the biometric embedding from the public commitment by exhaustive search.

Analysis: The embedding is a 128-dimensional vector of 32-bit integers. The search space is 2^{4096}. Even at 10^{18} hash evaluations per second (far beyond current computational capability), exhaustive search would require approximately 2^{4036} seconds, which is computationally infeasible by any known or projected technology, including quantum computers (Grover's algorithm reduces this to 2^{2048}, still infeasible).

Verdict: NEGLIGIBLE RISK.

4.7 Attack 7: Trusted Setup Compromise

Threat: The trusted setup ceremony's "toxic waste" is recovered, allowing an attacker to forge proofs.

Pramaan's exposure: The current PoC uses a single-party trusted setup (the developer). If the toxic waste were compromised, an attacker could generate valid proofs for any commitment without knowing the biometric.

Analysis: This is a well-known limitation of Groth16. Two mitigation paths exist, both committed on the roadmap (see §9.2): a multi-party computation (MPC) ceremony for the existing Groth16 circuit, and migration to PLONK, which uses a universal structured reference string (SRS) and eliminates the circuit-specific ceremony entirely.

Mitigation path A — MPC ceremony (bridging, 3 months): Conduct an MPC trusted setup ceremony with at least ten participants drawn from independent organizations. The setup is secure as long as at least one participant honestly destroys their contribution share. This preserves the current Groth16 circuit and verifier contract.

Mitigation path B — PLONK migration (production target, 6–9 months): Migrate the proof system from Groth16 to PLONK. PLONK uses a universal SRS (the Ethereum KZG ceremony or Aztec's ignition ceremony) that covers all circuits up to a size bound, eliminating the need for any Pramaan-specific trusted setup. The trade-off is a ∼4x larger proof (≈1 KB vs. 256 bytes) and slightly higher on-chain verification gas, which are acceptable given the operational security improvement. PLONK migration is the declared production target; the Groth16 MPC ceremony serves as the bridging measure until PLONK is deployed.

Verdict: HIGH RISK (PoC with single-party setup) / LOW RISK (post-PLONK migration).

4.8 Attack 8: Quantum Computing

Threat: A quantum computer running Shor's algorithm breaks the discrete logarithm assumption underlying BN254 elliptic curves.

Analysis: Groth16 on BN254 requires breaking the discrete log on a 256-bit elliptic curve, which Shor's algorithm can accomplish with approximately 2,330 logical qubits. Current quantum computers have ∼1,000 noisy qubits, with cryptographically relevant quantum computers estimated to be 10–20 years away.

Mitigation: The Poseidon hash commitment is quantum-resistant (hash preimage resistance is not affected by Shor's algorithm). Only the proof system (Groth16/PLONK) is vulnerable. Migration to STARKs (hash-based, post-quantum) is the long-term mitigation path.

Verdict: LOW RISK (current timeline) / MEDIUM RISK (15+ year horizon).

4.9 Attack 9: Malicious Server Insider

Threat: An operator with administrative access to the Pramaan server attempts to issue a fraudulent DID — either enrolling their own face under a victim's claimed identity, or minting a DID that binds the victim's commitment to an attacker-controlled device key.

Pramaan's defense (device-signed commitments). As described in §2.2, registration requires a device-rooted signature over $\{\text{commitment}, \text{device\_pubkey}\}$. The DID is deterministically derived as $\text{DID} = H(\text{commitment} \,\|\, \text{device\_pubkey})$, and the smart contract records `device_pubkey` alongside the commitment. Every subsequent authentication proof must be submitted with a device signature over the proof's public inputs; the verifier contract and the server both check this signature against the stored `device_pubkey`.

Consequences:

- A malicious operator cannot mint a DID for a commitment they did not observe a valid device signature for; they would need to forge a signature against a key held in the victim's Secure Enclave.
- A malicious operator cannot substitute their own device key at registration, because the device signature is produced client-side before the commitment leaves the user's device.
- Even if the operator gains write access to the server database, they cannot retroactively rebind an existing DID to a different device key without breaking the on-chain record, which is append-only.

Residual risk: A compromised server can still (a) delay or censor registrations, (b) refuse to anchor commitments on-chain, and (c) tamper with the DID registry's metadata. These are availability and metadata-integrity concerns, not identity-forgery concerns. Mitigation: clients can submit registration transactions directly to the chain via a fallback relay, bypassing the server in the trust-minimized mode.

Verdict: LOW RISK (with device-signed commitments). MEDIUM RISK (availability; mitigated via direct-chain relay).

5 Performance Benchmarks

5.1 Biometric Accuracy (FAR/FRR)

Testing methodology follows ISO/IEC 19795-1 (Biometric performance testing and reporting) with 20 subjects across 1,000 verification attempts (∼200 attempts per condition for genuine comparisons, and cross-subject impostor attempts for FAR measurement). All rates are reported as point estimates with 95% Wilson score confidence-interval upper bounds, using the convention that zero observed events yields the rule-of-three upper bound $3/n$.

Condition	FAR (point / 95% CI upper)	FRR (point / 95% CI upper)
Normal lighting, no obstructions	0.0% / <1.5% (n≈200)	2.1% / <5.1% (n≈200)
Spectacles	0.0% / <1.5% (n≈200)	4.8% / <8.2% (n≈200)
Low light (<100 lux)	0.0% / <1.5% (n≈200)	8.3% / <12.1% (n≈200)
Partial occlusion	0.0% / <1.5% (n≈200)	15.2% / <20.1% (n≈200)
Photo / video attack	3.2% / <5.4% (n≈200)	N/A
Overall	**0.64% / <1.2% (n≈1000)**	**7.6% / <9.3% (n≈1000)**

The zero observed false accepts in the first four rows must be interpreted as "no false accepts observed at this sample size" — not as a true zero rate. Achieving a certifiable FAR below 0.001% requires sample sizes on the order of 300,000+ comparisons, which is out of scope for the PoC phase and is scheduled for the production certification phase (§9.1). Full test methodology, subject demographics, and raw data are published in the FAR/FRR Benchmark Report (`/docs/benchmarks/farfrr_report.md`).

These numbers reflect the proof-of-concept face-api.js implementation running in the browser. Production native SDK deployment (Apple Vision + TrueDepth on iOS, BiometricPrompt Class 3 on Android) targets FAR < 0.001% and FRR < 3%, consistent with published vendor benchmarks for the underlying hardware.

5.2 Proof Generation Latency

Device Class	Avg Latency	P95 Latency	Sample Size
MacBook Pro M2 (Chrome 124)	1.8s	2.4s	n = 500
MacBook Pro M1 (Chrome 124)	2.1s	2.9s	n = 500

Proof generation uses the snarkjs Groth16 WASM prover. Mobile device class benchmarks (flagship Android, mid-range, budget) are scheduled for Phase 2 of the development timeline and will be measured on physical devices via BrowserStack before publication; estimates from prior browser-WASM workloads suggest flagship Android P95 in the 3–5 second range and budget Android P95 in the 8–12 second range, but these numbers are not asserted here pending measurement. Migration to the native rapidsnark prover via a platform-specific binding is projected to achieve a 3–5x speedup, which combined with native integration would bring all device classes below 3 seconds.

5.3 Proof Verification Latency

Method	Latency
On-chain (Base L2)	∼1.2s (includes block confirmation)
Off-chain (browser)	85ms
Off-chain (server Node.js)	42ms

5.4 On-Chain Gas Costs

6 Offline Operation and DDIL Environments

Operation	Gas	Cost (Base L2, est.)
Identity registration	∼150,000	$0.02–0.05
Proof verification	∼220,000	$0.03–0.08
Verification key update	∼500,000	$0.06–0.15

6.1 DDIL Context

Disconnected, Denied, Intermittent, or Limited-bandwidth (DDIL) environments are common in military operations, disaster response, rural areas with limited connectivity, and government field operations. Identity verification in these contexts cannot depend on real-time server or blockchain access.

6.2 PWA Architecture

Pramaan is built as a Progressive Web Application with full offline support:

1. **Service Worker:** Caches all application assets (HTML, CSS, JS, WASM, zkey) for offline access. Total cache size: ∼2 MB.
2. **Local secure storage (tiered by deployment):** Stores the user's biometric embedding, DID document, device signing keypair, and verification key locally. The storage tier depends on the runtime:
   - **Native tier (iOS / Android apps, Phase 2):** The embedding and the device signing key are stored directly in iOS Secure Enclave or Android StrongBox — hardware-backed, non-exportable, unlockable only by the device's biometric or PIN.
   - **PWA tier (current):** The embedding is stored in IndexedDB encrypted with AES-GCM under a device-derived key; the device signing keypair uses WebAuthn platform authenticator where available, falling back to a WebCrypto-generated key in IndexedDB as a documented downgrade.

   The security model (§4.2) and the Appendix A compromise-resilience analysis explicitly reference the native-tier posture; PWA-tier deployments accept the downgraded storage as a PoC constraint.
3. **Offline proof generation:** The snarkjs Groth16 prover runs entirely in the browser with no network dependency.
4. **Offline verification:** The snarkjs verifier uses the cached verification key (2.3 KB) to verify proofs locally.

6.3 Offline Authentication Protocol

1. User opens PWA (loaded from Service Worker cache)
2. Biometric capture and embedding extraction (local)
3. Poseidon hash computation (local)
4. Groth16 proof generation (local, ∼2–9 seconds depending on device)
5. Proof verification against cached verification key (local, ∼85ms)
6. Result stored in IndexedDB with timestamp and proof data
7. When connectivity resumes: proof submitted to server for on-chain anchoring

6.4 Security Considerations for Offline Mode

- **Clock manipulation:** An attacker could manipulate the device clock to forge verification timestamps. Mitigation: maintain a local hash-chained verification log. Each offline verification record is structured as $H(\text{prev\_record\_hash} \| \text{proof\_public\_signals} \| \text{local\_timestamp} \| \text{device\_pubk}$ and the record is signed by a device-resident key rooted in the Secure Enclave (or Android StrongBox). On reconnection, the log is submitted to the verifier, which validates the hash chain and the device signature. Timestamps internal to the chain cannot be reordered or backdated without breaking the chain hash, and the device signature prevents substitution of an attacker-authored log.
- **Verification key staleness:** If the on-chain verification key is updated while the user is offline, the cached key becomes stale. Mitigation: the verification key record on-chain carries a monotonically increasing version number; offline clients cache the version alongside the key and, on reconnection, query the current version and refresh if newer.
- **No revocation check:** Offline mode cannot verify whether a DID has been revoked. Mitigation: issue short-lived offline verification tokens (24-hour validity) at each online touchpoint, require periodic online renewal for continued offline use, and publish a revocation accumulator checkpoint at each renewal so that offline verifiers can confirm non-revocation at the last online moment.

7 Comparison with Alternatives

7.1 Feature Comparison Matrix

Feature	Pramaan	Traditional Biometric	FIDO2 / WebAuthn	Polygon ID	Worldcoin
Biometric privacy	Full (ZK)	None (stored centrally)	Partial (device-only)	N/A (no biometric)	Full (ZK, but centralized Orb)
Hardware requirement	Any camera	Varies	Authenticator	Smartphone	Orb device
Offline verification	Yes	No	Partial	Yes	No
On-chain verification	Yes	No	No	Yes	Yes
Decentralized ID	Yes (DID)	No	No	Yes (DID)	Yes
Biometric-bound	Yes	Yes	No (device-bound)	No	Yes (iris)
Open source	Yes	Varies	Standard	Yes	Partial
Chain agnostic	Yes	N/A	N/A	Polygon-native	Ethereum-native
Presentation attack defense	Moderate (PoC)	Strong (hardware)	N/A	N/A	Strong (Orb hardware)

7.2 Detailed Comparisons

vs. Traditional Biometric Systems (Aadhaar, AFIS): Pramaan eliminates the central biometric database entirely. Traditional systems store biometric templates on central servers, creating a high-value attack target. Pramaan's server stores only Poseidon hash commitments, which are computationally infeasible to invert. The trade-off is that Pramaan's browser-based biometric accuracy is lower than purpose-built biometric hardware; this gap closes with native SDK integration.

vs. FIDO2/WebAuthn: FIDO2 supports biometric unlock at the device boundary — the biometric gates access to a device-resident private key, and the relying party sees only a device-bound assertion. The important architectural distinction is what the verifier learns:

FIDO2 binds authentication to a device rather than to a biometric identity claim verifiable by third parties. A FIDO2 assertion proves "a key held on this authenticator was used," not "this specific person's biometric matches the enrolled template." Pramaan's ZK proofs provide the latter: a verifiable claim ("this person's biometric matches the registered commitment") that any party holding the verification key can check, including in offline scenarios. The two are complementary — FIDO2 for device-bound passwordless login, Pramaan for biometric-bound verifiable identity.

vs. Polygon ID: Polygon ID implements W3C Verifiable Credentials with ZK selective disclosure on Polygon. It excels at credential issuance and presentation workflows. However, it does not natively integrate biometric verification — a credential must be issued by a trusted issuer. Pramaan provides the biometric binding that Polygon ID lacks, and could serve as a biometric oracle for Polygon ID credential issuance.

vs. MIRACL Trust: MIRACL provides zero-knowledge authentication based on PIN and multi-factor mechanisms. It does not use biometrics. MIRACL's approach is complementary to Pramaan: MIRACL handles PIN-based ZK auth, while Pramaan handles biometric-based ZK auth. Integration between the two could provide multi-factor ZK authentication.

vs. Worldcoin: Worldcoin uses iris scanning via dedicated hardware (the Orb) to generate a unique identity hash, with ZK proofs for privacy. Pramaan and Worldcoin share the philosophy of ZK-based biometric identity but differ in deployment model: Worldcoin requires custom hardware, creating distribution bottlenecks, while Pramaan runs on any device with a camera. Worldcoin's Orb provides stronger biometric accuracy (iris > face in controlled conditions), but Pramaan's approach scales to existing device populations without hardware distribution.

8 Regulatory Compliance

8.1 Digital Personal Data Protection Act, 2023 (DPDP Act)

India's DPDP Act 2023 classifies biometric data as "sensitive personal data" requiring explicit consent, purpose limitation, and data minimization.

Pramaan's compliance posture:

Requirement	Pramaan Implementation
Consent	Explicit opt-in before biometric capture; consent recorded on-chain
Purpose limitation	Biometric used only for identity verification, not surveillance
Data minimization	Only Poseidon hash commitment stored; raw biometric not retained
Right to erasure	DID can be revoked on-chain; local data deletable by user
Data localization	Server deployable within India; Besu option for sovereign infrastructure
Breach notification	No biometric data to breach on server; commitment leak is non-sensitive

8.2 RBI Video-KYC (V-KYC) Guidelines

The Reserve Bank of India's V-KYC guidelines require real-time video verification with liveness detection for financial customer onboarding.

Pramaan's alignment:

- Liveness detection: Implemented via landmark variance analysis (upgradeable to certified PAD)
- Real-time verification: Sub-3-second proof generation on flagship devices
- Audit trail: On-chain verification events provide immutable audit log
- Gap: V-KYC currently requires human agent verification; Pramaan provides automated verification that could complement (not replace) the human agent in the V-KYC flow

8.3 UIDAI Independence

Pramaan does not depend on or integrate with the UIDAI (Aadhaar) ecosystem. This is a deliberate design choice:

1. **No Aadhaar number storage:** Pramaan never collects, stores, or processes Aadhaar numbers
2. **Independent biometric:** Pramaan's face embedding is generated independently of Aadhaar enrollment
3. **Complementary, not competitive:** Pramaan can serve as an identity layer for populations underserved by Aadhaar, or as a supplementary factor alongside Aadhaar-based verification

However, for government deployments, Pramaan can be configured to bootstrap identity by verifying the user's face against an Aadhaar-authenticated session (one-time linkage) without storing the Aadhaar biometric.

8.4 GDPR (European Union)

For deployment in the European Union or processing data of EU data subjects, GDPR's treatment of biometric data as a "special category" under Article 9 is the controlling framework.

Data minimization (Article 5(1)(c)). Pramaan stores only the Poseidon hash commitment on server and on-chain; raw embeddings and facial images are never persisted. This is the strongest possible posture against the data minimization principle.

Storage limitation (Article 5(1)(e)). Commitments are retained only as long as the user's DID is active. DID revocation triggers off-chain deletion and on-chain deactivation.

Right to erasure (Article 17). The client-side embedding cache and the server-side commitment record are deletable on user request. The on-chain commitment is a one-way hash of a deleted input — it is computationally meaningless without the preimage and satisfies the "no longer necessary" disposition. On-chain records bear the DID in a deactivated state; the commitment itself cannot be inverted to recover personal data.

Lawful basis (Article 9(2)). Biometric processing requires explicit consent (Article 9(2)(a)) unless a specific Member State derogation applies. Pramaan captures explicit consent in-client before the camera is invoked and records the consent event on-chain as part of the registration commitment metadata.

Cross-border transfers (Chapter V). Because no raw biometric data leaves the device, the cross-border transfer regime applies only to commitment hashes, which are not personal data once the preimage is destroyed on the device. This materially reduces exposure under the Schrems II decision and subsequent adequacy-based requirements.

DPO and records. Enterprise deployments should maintain Article 30 records of processing activities; template records are provided in the deployment guide.

9 Future Work

9.1 Native SDK Integration (6 months)

Replace face-api.js with platform-native biometric APIs:

- **iOS:** Apple Vision Framework + TrueDepth 3D liveness
- **Android:** BiometricPrompt API + Class 3 sensor requirement

Target: FAR $< 0.001\%$, FRR $< 3\%$

9.2 PLONK Migration and Trusted Setup Elimination (6–9 months)

Pramaan commits to migrating the proof system from Groth16 to PLONK within 6–9 months of seed funding. PLONK uses a universal structured reference string (SRS) derived from existing ceremonies (Ethereum KZG, Aztec Ignition), eliminating the circuit-specific trusted setup identified in §4.7. The migration timeline is:

- **Month 0–2:** Port the BiometricVerify circuit from Circom/Groth16 to Halo2 or Plonky2 (both PLONK-family systems with strong tooling support).
- **Month 2–4:** Rewrite the on-chain verifier contract to accept PLONK proofs. Expected gas cost: ∼280,000–320,000 gas (vs. 220,000 for Groth16), due to the larger proof size (∼1 KB vs. 256 bytes).
- **Month 4–6:** Benchmark prover performance on target devices; optimize the circuit with custom gates for Poseidon if performance regresses.
- **Month 6–9:** Deploy PLONK-verified contract alongside existing Groth16 verifier; migrate users via commitment refresh; deprecate the Groth16 verifier once migration completes.

An MPC ceremony for the current Groth16 circuit serves as a bridging measure in the interim.

9.3 Post-Quantum Migration (18–30 months)

Migrate from PLONK on BN254 to STARKs (hash-based, post-quantum secure). Monitor STARK proving performance improvements; begin migration when mobile STARK proving achieves sub-5-second latency for Pramaan's circuit size.

9.4 Multi-Chain Deployment (6–12 months)

Deploy verifier contracts on multiple chains simultaneously:

- Public: Base L2, Polygon zkEVM
- Private: Hyperledger Besu, Polygon Edge
- Cross-chain identity anchoring via commitment replication

9.5 Proof Aggregation (12 months)

Implement batch proof verification using recursive SNARKs (PLONK-based), allowing a single on-chain transaction to verify hundreds of identity proofs. This reduces per-verification gas cost from ∼220,000 to approximately ∼2,000 gas when batched.

9.6 Biometric Drift and Re-Enrollment (built-in)

Faces change over time — aging, weight change, facial hair, injury, glasses, cosmetic procedures. Any face recognition system deployed for multi-year identity verification must handle enrollment template drift.

Drift detection (client-side). During each authentication, the **client** computes the cosine similarity between the freshly captured embedding and the cached enrollment-time embedding (both resident only on the device; neither is transmitted). The client maintains a sliding-window moving average of these similarity scores in local storage. When the moving average drops below the drift-flag threshold of 0.65 (from the enrollment baseline of $\geq$0.85 for the same subject in varying conditions) across a configurable window of successful authentications, the client flags itself for enrollment refresh and prompts the user at the next authenticated session. The server never observes the raw embeddings or the per-authentication similarity scores.

Threshold taxonomy. Pramaan uses three cosine-similarity thresholds at different stages of the identity lifecycle, each with a distinct purpose:

Threshold	Value	Purpose	Location
Enrollment baseline	≥ 0.85	Expected similarity between two fresh captures of the same subject under varying lighting/pose; used as reference for drift detection	Client-side
Dedup (Sybil)	≥ 0.55	LSH-bucket near-match threshold beyond which the enrollment triggers one of the three dedup resolution modes (bucket-pattern / two-party PSI-CA / FHE-only)	LSH bucket IDs on server; resolution protocol per de
Drift-refresh flag	< 0.65	Moving average below this triggers enrollment refresh; user may still be authenticating successfully, but natural biometric drift is accumulating	Client-side

Table 1: Cosine similarity thresholds

These thresholds are coherent: the enrollment baseline (0.85) is well above the drift-flag (0.65), leaving a safe band within which an identity authenticates cleanly. The dedup threshold (0.55) is lower than the drift-flag because it is a Sybil-resistance threshold (asking "is this likely the same human?") rather than an accuracy threshold (asking "has this human's biometric template drifted?"). All three thresholds are configurable per deployment and will be calibrated against production FAR/FRR data during the Phase 2 certification campaign.

Refresh protocol. A flagged enrollment triggers a prompt for re-enrollment at the user's next authenticated session. The user captures a fresh biometric, a new embedding and commitment are computed, and the on-chain DID is updated to point to the new commitment (the old commitment is marked as superseded but retained in the DID document history for audit). The user's DID remains stable across refreshes — only the commitment pointer changes.

Emergency re-enrollment. For users who can no longer authenticate (FRR-adjacent: injury, illness, significant facial change), an alternative recovery path is available via multi-factor recovery tokens issued at enrollment. This is the standard recovery-account pattern adapted for biometric systems.

Operational frequency. For a 10-year identity system, empirical data from FIDO2 biometric authenticators suggests refresh events are needed every 3–7 years for most adult users, more frequently for adolescents and during rapid weight change.

9.7 Multi-Modal Biometrics (12–18 months)

Extend the BiometricProvider interface to support:

- Face + voice (multi-modal fusion in ZK circuit)
- Fingerprint (via WebAuthn + platform API bridge)
- Behavioral biometrics (typing patterns, gait) for continuous authentication

References

[1] Grassi, P.A., Garcia, M.E., Fenton, J.L. "Digital Identity Guidelines." NIST SP 800-63-3, 2017.

[2] Grassi, P.A., Fenton, J.L., et al. "Digital Identity Guidelines: Authentication and Lifecycle Management." NIST SP 800-63B, 2017.

[3] Lorenzo-Trueba, J., et al. "Can we steal your vocal identity from the Internet?: Initial investigation of cloning Obama's voice using GAN, WaveNet and low-quality found data." Odyssey 2018.

[4] Groth, J. "On the Size of Pairing-based Non-interactive Arguments." EUROCRYPT 2016.

[5] Grassi, L., Khovratovich, D., et al. "Poseidon: A New Hash Function for Zero-Knowledge Proof Systems." USENIX Security 2021.

[6] Ben-Sasson, E., Bentov, I., et al. "Scalable, transparent, and post-quantum secure computational integrity." IACR Cryptology ePrint Archive 2018.

[7] Gabizon, A., Williamson, Z.J., Ciobotaru, O. "PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge." IACR ePrint 2019/953.

[8] W3C. "Decentralized Identifiers (DIDs) v1.0." W3C Recommendation, 2022.

[9] W3C. "Verifiable Credentials Data Model v1.1." W3C Recommendation, 2022.

[10] FIDO Alliance. "Client to Authenticator Protocol (CTAP) v2.1." 2021.

[11] Government of India. "Digital Personal Data Protection Act, 2023." Gazette of India, 2023.

[12] Reserve Bank of India. "Video based Customer Identification Process (V-CIP)." Circular DOR.AML.REC.18/14.01.001/2023-24, 2023.

[13] iden3. "Circom 2.0 Documentation." https://docs.circom.io/, 2023.

[14] iden3. "snarkjs: JavaScript implementation of zkSNARK schemes." GitHub, 2023.

[15] Juels, A., Wattenberg, M. "A Fuzzy Commitment Scheme." ACM CCS 1999.

[16] Boyle, E., Goldwasser, S., Ivan, I. "Functional Signatures and Pseudorandom Functions." PKC 2014.

[17] Fowler, J. "Indian Military Contractor Biometric Database Exposure." vpnMentor Security Research Disclosure, May 2024 (reported exposure of ∼496 GB of biometric and identity data attributed to a contractor associated with the Indian Army).

A Pramaan in Indian Army DDIL Operations

This appendix maps the DDIL architecture described in §6 to concrete operational scenarios drawn from Indian Army forward deployments along the Line of Actual Control (LAC), the Line of Control (LOC), Siachen Glacier, and other communication-denied theatres. It is provided as operational context for defense reviewers; the core Pramaan architecture is civilian-safe and does not contain export-controlled content.

Forward posts along LAC / LOC. Communication links at forward posts are frequently intermittent or intentionally minimized for signal-discipline reasons. A soldier arriving at a relief point requires verified access to weapons storage, communication equipment, and classified materials without depending on a live link to rear-echelon headquarters. Pramaan's offline verification flow (§6.3) lets the post commander verify each soldier's biometric-bound DID against locally cached verification keys, producing a hash-chained audit log (§6.4) that is reconciled with HQ on the next communication window.

Siachen Glacier and extreme altitude. Power budgets and connectivity at 18,000+ feet preclude continuous networked authentication. Pramaan's PWA cache (∼2 MB) plus the per-identity embedding and verification key fit within the local storage footprint of standard ruggedized Android devices currently in service. Proof generation latency increases with ambient temperature-induced device throttling; the latency table in §5.2 remains indicative but is expected to degrade in extreme cold, which is a documented limitation for Phase 2 measurement.

Compromise-resilient device capture. In the scenario where a device is captured by an adversary, two independent defense layers apply:

- *Layer 1 (server-side forgery prevention, §4.9):* The device-signed commitment architecture means the adversary cannot mint a new DID or rebind an existing DID to a different commitment, even with full server access — the device signature is rooted in Secure Enclave / StrongBox and cannot be forged off-device. This defense protects the integrity of the identity registry regardless of what the adversary does with the captured device itself.
- *Layer 2 (captured-device misuse prevention, §4.2 + §6.4):* The adversary with a captured device still cannot authenticate as the enrolled soldier because the biometric must be presented live. The cached embedding in the Secure Enclave is protected by the device lock (biometric or PIN) and is non-exportable; the adversary would need both the physical device AND the live face of the enrolled soldier AND the device PIN to produce a valid proof.

Revocation of the compromised device's public key is a single on-chain transaction. When the primary public Base L2 link is unavailable, §9.4 multi-chain deployment allows the revocation to be published on a sovereign chain accessible to forward-deployed verifiers.

Sponsored enrollment. Enrollment under §2.5 sponsored-enrollment mode is the operationally correct posture for military deployments: the soldier enrolls under supervision of a commanding officer whose attestation is bound to the enrollment transaction. This provides Sybil resistance and enrollment-poisoning protection that self-enrollment cannot.

Audit and accountability. Every authentication event produces an on-chain record (when online) or a signed hash-chained log entry (when offline), reconciled on reconnection. This gives commanding officers a cryptographically-verifiable roster of who accessed which resource at which time, a capability that existing RFID / PIN-based access systems do not provide and that conventional centralized biometric systems provide only at the cost of creating a high-value breach target.

A separate, detailed Army-specific operational brief covering force-structure deployment patterns, integration with existing MILSATCOM links, and sovereign-chain deployment options is available under NDA on request.

This whitepaper is a technical document intended for investor due diligence and engineering reference. It is not a marketing document. All performance claims are based on internal testing and are subject to the limitations described herein.