Opalite Love Whitepaper

version 0.1

Swipe right, reveal later.

Privacy-Preserving Dating on Midnight Network

July 2025

Opalite Love Team

https://opalite.love

Table of Contents

1. Overview

Opalite Love is a privacy-first dating app where zero-knowledge proofs protect user identity, hide individual swipe decisions, and verify trust attributes — all without exposing underlying personal data. Unlike conventional dating apps that store plaintext profiles and swipe histories on centralized servers, Opalite Love uses Midnight Network's ZK-native smart contracts to ensure that personal data never appears on-chain, individual likes remain hidden unless both users match, and trust/safety checks (age, uniqueness, reputation) are proven cryptographically rather than self-attested.

2. How Midnight & ZK Are Used

2.1 ZK-Based Age Verification

What stays private: The user's actual birthdate, government ID, and full identity.

What is proven: A boolean — "this user is 18 or older."

A user completes identity verification with a trusted attestation provider. The provider issues a signed attestation containing the user's verified birthdate. The user's device generates a ZK proof locally that takes the attested birthdate as a private witness and the current date as a public input, outputting only is_over_18: true or false. This proof is submitted to a Midnight smart contract. The contract validates the ZK proof and the attestation signature — but the birthdate itself never appears on-chain, never leaves the device, and is never visible to other users or the platform.

2.2 Proof of Uniqueness

What stays private: The user's underlying identity.

What is proven: "This user has not already registered an account."

On registration, the user derives an identity commitment from their attested identity. This commitment is submitted to a Midnight smart contract as a nullifier. The ZK proof verifies that the commitment was derived from a valid, attested identity (private input) and that the commitment has not been seen before on-chain. This prevents duplicate/bot accounts without revealing who the user is.

2.3 Privacy-Preserving Location Range

What stays private: The user's exact GPS coordinates.

What is proven: "This user is within X km of that user."

Each user submits their approximate location as a geohash or grid cell ID (not raw coordinates). A ZK circuit takes both users' grid cells as private inputs and outputs only within_range: true or false. Users see potential matches as "within 25 km" — never the exact distance or location.

2.4 Mutual Matching with Hidden Likes

What stays private: Individual swipe decisions. No user can see who liked them unless there is a mutual match.

What is proven: "Both users swiped right on each other."

Step 1 — Swipe commitment: When User A swipes right on User B, A's device generates a ZK proof that commits to "A likes B" using a nullifier derived from both user IDs. This commitment is submitted to a Midnight smart contract. The on-chain record reveals nothing — it's an opaque hash.

Step 2 — Match verification: When User B swipes right on User A, B's device generates a matching commitment. The Midnight smart contract's ZK circuit checks whether both commitments correspond to the same pair (A, B) and both are "like" commitments. If true, the contract emits a Match event containing only a shared match ID.

Step 3 — Match notification: Both users are notified via push notification. The match proof unlocks a shared encryption key (derived from the match circuit) that both users can use to decrypt each other's profiles and initiate encrypted chat.

2.5 Private Reporting & Reputation

What stays private: The reporter's identity and the specific evidence.

What is proven: "This report is from a verified, unique user and meets the threshold for review."

When a user reports another user, they submit a ZK proof to a reputation smart contract. The proof verifies the reporter is a registered, unique user, prevents duplicate reports via a nullifier, and matches a valid policy violation type. The contract accumulates reputation signals privately. If a user accumulates enough verified reports, their reputation score drops below a threshold, and they are flagged or hidden.

3. The Identity Commitment Process

The goal is to create a unique, deterministic fingerprint of a user's verified identity that cannot be linked back to their real-world identity, can be verified inside a ZK circuit, and prevents the same person from registering multiple accounts.

3.1 Step 1: The KYC Attestation (Off-Chain)

The user completes identity verification with a trusted KYC provider. The provider verifies their government ID and returns a signed attestation containing their verified identity data and a cryptographic signature. This attestation is stored locally on the user's device and never goes on-chain.

3.2 Step 2: The Secret Trapdoor (On-Device)

The user's device generates a random secret number called a "trapdoor" or "nullifier secret". This is stored securely on-device in the secure enclave. It never leaves the device and never appears on-chain. Without it, if the same verified identity data were used on another ZK-based platform, the resulting commitment would be identical, allowing cross-platform tracking. The trapdoor ensures the commitment is unique to this user on this app.

3.3 Step 3: The Pedersen Hash (Inside the ZK Circuit)

When the user registers, their device generates a ZK proof. Inside the circuit, two operations occur:

- Signature verification: The circuit verifies the KYC provider's signature is valid for the identity data.
- Commitment computation: The circuit computes identity_commitment = PedersenHash(verified_identity_data, secret_trapdoor).

A Pedersen Hash generalizes the concept of a Pedersen Commitment from 2 inputs to n inputs. It maps an arbitrary-length message to a single elliptic curve point: H(m) = m0*G0 + m1*G1 + m2*G2 + ... + mn*Gn. The generators (G0, G1...) are predefined and publicly known, with no known discrete log relationship between them. Standard hashes like SHA-256 require tens of thousands of arithmetic constraints in ZK circuits, while Pedersen Hashes require only ~1,000-1,500 constraints because they use elliptic curve scalar multiplication which maps naturally to field arithmetic.

3.4 Step 4: On-Chain Nullifier Check

The ZK proof and the public identity commitment are submitted to a Midnight smart contract. The contract verifies the ZK proof and checks if the commitment already exists in its registry. If not, the commitment is added and registration succeeds. If yes, registration is rejected. The blockchain only records the opaque commitment and the fact that a valid proof was verified — no names, photos, or demographics.

4. Architecture: Where Data Lives

Figure: System architecture. The Mobile App (client) generates ZK proofs and submits them to Midnight Network. Profile data is stored encrypted on IPFS/Filecoin, and chats flow through Nostr relays.

4.1 On-Chain (Midnight Network)

- Identity commitments (nullifier hashes — no PII)
- Age verification proof results (boolean only)
- Swipe commitments (opaque hashes)
- Match events (shared match ID only — no participant identities)
- Reputation contract state (aggregate scores — no report details)
- Content hash references to encrypted profile blobs (IPFS CIDs)

All on-chain data is either a ZK proof output, a hash/commitment, or a boolean flag. No plaintext personal data appears on-chain.

4.2 Off-Chain (Encrypted Storage)

Encrypted profile data (photos, bio, demographics): Stored on IPFS with Filecoin as a pinning/storage option for persistence guarantees. Only the CID is referenced on-chain. Upon mutual match, the match circuit derives a shared key that unlocks the counterparty's profile for decryption.

Encrypted chat messages: Chat uses Nostr (Notes and Other Stuff Transmitted by Relays) for decentralized, censorship-resistant messaging. Messages are end-to-end encrypted using NIP-04 or NIP-17 standards, with encryption keys derived from the match proof. Messages are stored on Nostr relays as ciphertext — neither the relays, the platform, nor Midnight validators can read chat content.

4.3 Mobile App (Client-Side)

- ZK proof generation runs on-device using WASM-compiled circuits

• Wallet integration via Midnight wallet SDK

• Local key management for profile/chat encryption

• Nostr keypair management for encrypted direct messages

• Push notifications for match alerts

5. Why This Is Midnight-Native

1. ZK proofs are first-class, not bolted on. Age verification, uniqueness, location range, and mutual matching all use ZK circuits as the core mechanism. The proofs are validated by Midnight smart contracts, not by a trusted server.

2. Likes are hidden by default, not by policy. Conventional dating apps store likes in a database and promise not to show them. Opalite Love hides likes cryptographically — the blockchain records only opaque commitments. No server admin, no data breach, no subpoena can reveal who liked whom unless there is a mutual match.

3. Trust is proven, not self-attested. Age, uniqueness, and reputation are all ZK-verified on-chain. A user cannot lie about their age or create multiple accounts without breaking the ZK proof.

4. Decentralized attestation, not platform trust. Identity verification comes from independent KYC providers, and the proof is verified by Midnight's consensus — not by Opalite Love's servers. The platform cannot manipulate who passes verification.

5. Censorship-resistant matching. Match events are recorded on Midnight's blockchain. The platform cannot silently suppress matches, shadowban users, or manipulate the matching algorithm without it being visible on-chain.