

Decentralized Platforms: Comprehensive Comparative Analysis

Section 3 – Platform Deep Profile: Nostr
Version 3.0 | 16 August 2026 | Segment 03

3.1 Overview

Attribute	Value
Full name	Notes and Other Stuff Transmitted by Relays
Type	Minimal open social protocol
Primary identity	secp256k1 keypair (nsec / npub)
Transport	WebSocket to relays
Core object	Signed JSON event
Official NIPs repo	https://github.com/nostr-protocol/nips
Primary design goal	Censorship-resistant, simple, client-portable social communication

Nostr is deliberately minimal. It defines almost nothing beyond signed events and a simple request/publish model over WebSockets. Higher-level behaviour is layered via optional NIPs.

3.2 History and Design Philosophy

Nostr was proposed by fiatjaf around 2020–2021 as a reaction to the complexity of existing federated social protocols. The core insight: a social protocol can be reduced to cryptographic keypairs as identity, signed events as the only data type, relays that store/forward according to local policy, and clients that publish to and subscribe from any number of relays.

There is no global consensus, no blockchain, no required server-side account, and no canonical location for a user's data. Resilience comes from multi-relay publication and re-broadcasting.

- **Simplicity over completeness** – the base protocol stays tiny
- **Client freedom** – any client can implement any subset of NIPs
- **Cryptographic identity** – the keypair is the user
- **Relay non-trust** – relays can censor or drop events but cannot forge valid signatures

3.3 Architecture

```

##### WebSocket ##### Client ■ ##### Relay ■ - keys ■ EVENT / REQ / CLOSE ■ - store
■ - UI ■ EVENT / EOSE / OK ■ - filter ■ ■ - NIP ■ - policy ■ ##### publish to multiple relays ■
#####

```

Client – generates keypairs, creates and signs events, maintains subscriptions, renders UI, implements chosen NIPs. **Relay** – accepts events, validates signatures, stores events, serves them to matching subscribers. A user typically connects to several relays simultaneously.

3.4 Identity Model

Identity is a secp256k1 keypair. Private key (nsec) must be kept secret and is used to sign events. Public key (npub or hex) is the stable, permanent user identifier. There is no registration step — possession of the private key is control of the identity.

Bech32 encoding (NIP-19) is widely used: npub1... (public key), nsec1... (private key), note1... (event id), nevent1..., nprofile1...

3.5 Core Data Model – The Event (NIP-01)

3.5.1 Event Structure

```
{ "id": "<32-byte lowercase hex of SHA-256 of the serialized event>", "pubkey": "<32-byte lowercase hex of the public key>", "created_at": ,
  "kind": , "tags": [ "e", "", ... ], [ "p", "", ... ] , "content": "", "sig": "<64-byte lowercase hex of the Schnorr signature of the id>" }
```

3.5.2 Canonical Serialization for ID

id = SHA-256(JSON-array-serialization of [0, pubkey, created_at, kind, tags, content]). No extra whitespace; keys in exact order; UTF-8.

3.5.3 Python Implementation Example

```
import json, time, hashlib from coincurve import PrivateKey def create_event(private_key, kind, content, tags=None): if tags is None: tags = []
pubkey = private_key.public_key.format(compressed=True).hex() created_at = int(time.time()) ser = json.dumps([0, pubkey, created_at, kind, tags,
content], separators=(',', ':'), ensure_ascii=False) event_id = hashlib.sha256(ser.encode('utf-8')).hexdigest() sig =
private_key.sign(bytes.fromhex(event_id), hasher=None).hex() return {"id": event_id, "pubkey": pubkey, "created_at": created_at, "kind": kind,
"tags": tags, "content": content, "sig": sig}
```

3.5.4 Client ↔ Relay Messages

Client → Relay: ["EVENT", event] | ["REQ", sub_id, filter...] | ["CLOSE", sub_id]

Relay → Client: ["EVENT", sub id, event] | ["EOSE", sub id] | ["OK", event id, true/false, msg] | ["NOTICE", msg]

3.6 Important NIPs Beyond NIP-01

NIP	Purpose	Notes
NIP-01	Basic protocol	Mandatory
NIP-02	Contact / follow list	Kind 3
NIP-05	DNS-based identifiers	Mapping only; identity remains the key
NIP-07	window.nostr browser API	Common for web clients

NIP-09	Event deletion	Kind 5
NIP-17	Private Direct Messages	Preferred over NIP-04
NIP-19	bech32 encoding	npub, nsec, note, etc.
NIP-23	Long-form content	Kind 30023
NIP-25	Reactions	Kind 7
NIP-57	Lightning Zaps	Value-for-value payments

3.7 Client Ecosystem (mid-2026)

Mobile: Damus (iOS), Amethyst (Android), Primal. **Web:** Coracle, Iris, Snort, Primal Web. **Desktop:** Gossip, Pretty Good. Specialized clients for long-form, marketplaces, chat, and Lightning/Zaps also exist. Because identity is the keypair, users can move between clients without creating new accounts.

3.8 Current Status (August 2026)

Protocol is stable at the NIP-01 level. Large and still growing set of clients and relays. Active experimentation with new NIPs. Lightning Zaps have created a working value-for-value economy. Spam and discovery remain the largest unsolved UX problems. Relay operators continue to experiment with retention and moderation policies.

3.9 Strengths and Limitations

Strengths: extreme simplicity; highest client portability; strong censorship resistance; easy to implement; growing Lightning-native economy; no registration required.

Limitations: public content unencrypted by default; relays can censor; metadata visible to relays; spam/discovery unsolved at protocol level; not designed for private file storage or complex access control; multi-device key management requires care.

3.10 Comparative Position

Dimension	vs Peergos	vs ActivityPub	vs Matrix	vs SSB
Client diversity	Much higher	Comparable/higher	Comparable	Higher
Default privacy (public)	Weaker	Comparable	Weaker	Weaker
Identity portability	Highest	Lower	Lower	High
Implementation complexity	Lowest	Higher	Higher	Medium
Offline capability	Weak	Weak	Medium	Strong

Nostr is the clearest example in this report of a protocol-first, keypair-identity, event-log design optimized for client freedom and censorship resistance rather than private storage or strong default confidentiality of public posts.

3.11 Key Primary Sources

- NIPs repository: <https://github.com/nostr-protocol/nips>
- NIP-01: <https://github.com/nostr-protocol/nips/blob/master/01.md>
- Nostr protocol overview: <https://github.com/nostr-protocol/nostr>
- Readable NIP mirror: <https://nostr-nips.com>