B4OS

Private Messaging on Nostr: From NIP-04 to Marmot

Building unstoppable & secure communication

Javier Montoya

@jgmontoya
@JGMontoyaS

nostr

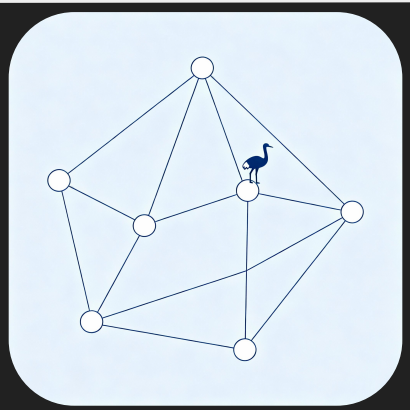
Under the Ban Hammer

CENTRALIZED = VULNERABLE

2025

- US House of Representatives, Russia, Nepal
- Vietnam, Russia, Nepal
- Nepal (more than 26+ apps banned)

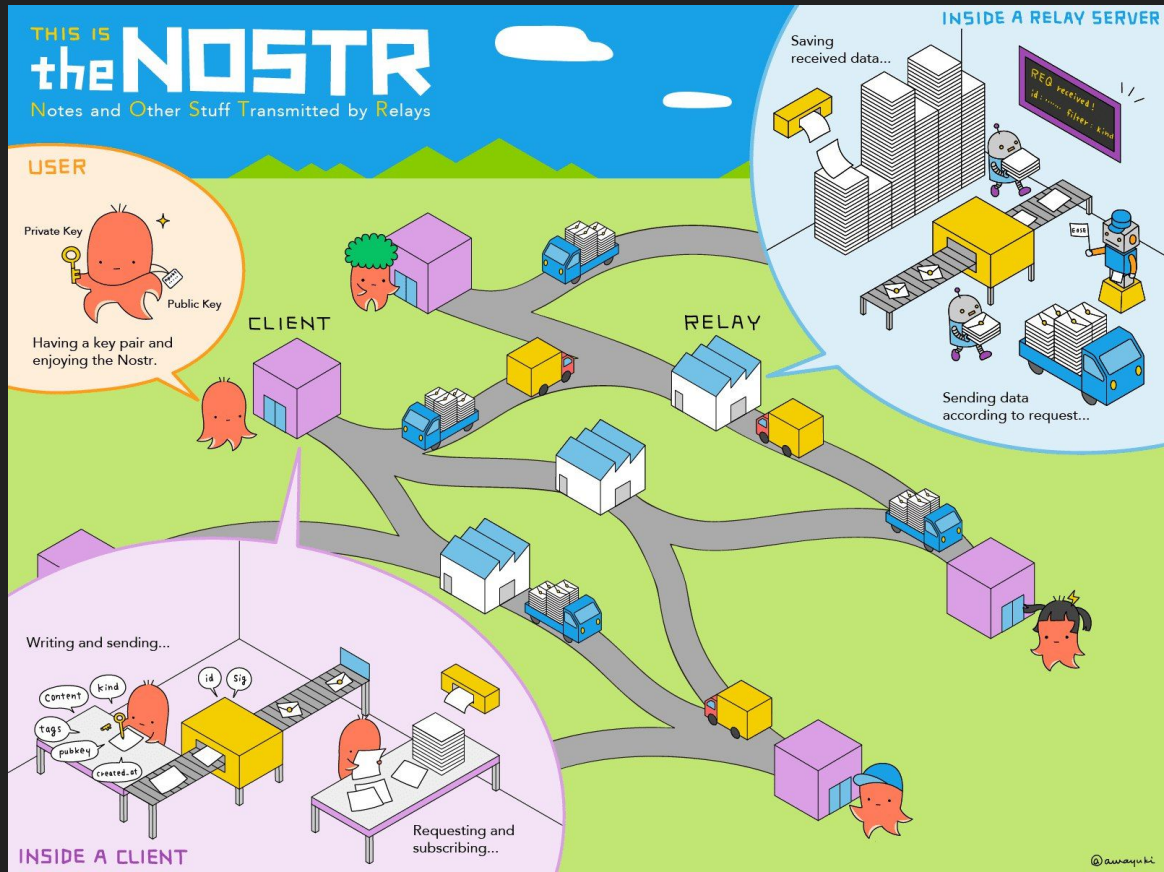
Freedom requires unstoppable communications

Centralized	Decentralized
	

nostr - Notes and Other Stuff Transmitted by Relays

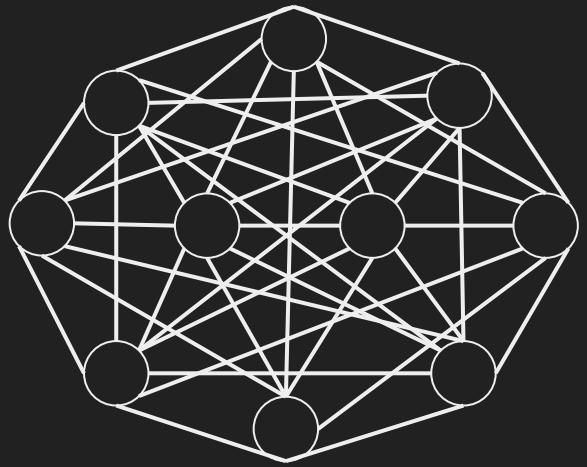
“The simplest open protocol that is able to create a censorship-resistant global "social" network once and for all.

It doesn't rely on any trusted central server, hence it is resilient; it is based on cryptographic keys and signatures, so it is tamperproof; it does not rely on P2P techniques, therefore it works.”



Nostr provides censorship resistance, but lacked secure group messaging

Why Group Messaging is Hard

Easy	Complex	Nearly impossible
		 x 10,000?
2 people	10 people	10,000 people

NIP-04: The Broken Beginning

- **Kind**: 4
- **Content**: base64-encoded, aes-256-cbc encrypted string of the message, encrypted using a shared cipher generated by combining the recipient's public-key with the sender's private-key
  - `"content": "<encrypted_text>?iv=<initialization_vector>"`
- **Tags**
  - MUST contain an entry identifying the receiver of the message
    - `["p", "<pubkey, as a hex string>"]`
  - MAY contain an entry identifying the previous message in a conversation or a message we are explicitly replying to
    - `["e", "<event_id>"]`

NIP-04: The Broken Beginning

- No group messaging
- Leaks metadata
  - Who is talking to who is made public
  - Message timestamps are public
  - Message frequency/patterns are observable
- No forward secrecy: if keys are compromised, all past messages can be read
- No post-compromise security: if keys are compromised, all future messages can be read
- And other problems
  - No versioning, no deniability, cryptographic weakness, etc…

NIP-17: Better But Limited

- Uses NIP-44 encryption and NIP-59 seals and gift wraps
- NIP-44 encrypted payloads provides stronger encryption
  - Introduces versioning, padding, HMAC-SHA256 for authentication
  - ChaCha20 instead of AES-CBC
- NIP-59 Gift Wraps provide a way to hide metadata
  - 3-layer encapsulation (rumor → seal → gift wrap)
  - Sender identity, timestamps, and content hidden from public

✅ What it solved: Strong encryption, metadata hiding, some group support

❌ What it didn't solve: Scalability (separate event per recipient), forward secrecy, post-compromise security

NIP-59: Gift Wraps 🎁

Rumor - **Unsigned** nostr event. If leaked, will be rejected by relays/clients and can't be authenticated. Provides deniability.

Seal - Signed event with encrypted rumor as content. Identifies the author without revealing content or recipient.

Gift Wrap - Signed (with ephemeral keys) event with encrypted seal as content. Can add metadata (recipient p tag) without revealing true author.

Enter MLS

MLS: Messaging Layer Security

An IETF internet standard for secure group messaging at scale

RFC 9420 • Peer-reviewed by cryptographers worldwide

Truly scalable: O(log n) instead of O(n) operations

This means that for a group of 10,000 people we need

~14 operations instead of 10,000

MLS Security Properties

- Message Confidentiality, Integrity and Authentication 🛡️
  - Full end-to-end encryption with cryptographic guarantees.

- Forward Secrecy 🔒⏪
  - If your key leaks today, yesterday's messages stay safe. They can't be decrypted.

- Post-Compromise Security 🩹
  - Even AFTER a key compromise, the protocol self-heals.

What MLS Needs

MLS is just the cryptography. It needs:

- Authentication Service 🔑
  - Verify member identities and manage keys
- Delivery Service 📨
  - Transport encrypted messages between members
- Storage (optional) 💾
  - Persist messages for offline/multi-device access

MLS + NOSTR = NIP-EE

MLS + NOSTR = Marmot

MLS provides the cryptographic foundation

Nostr provides the transport layer and the authentication

Marmot is the integration protocol, how they work together

How Marmot works

🎫 **KeyPackages**: Your "invitation card" - tells others how to add you.

- Advertise capabilities, provide credentials, and include signing keys for authentication.
- Users can publish them via Nostr relays or share directly between devices.

👋 **Welcome Events**: Provide everything new members need to join groups and start participating.

💬 **Group Events**: Messages that are sent within a group.

- Includes "control" events that update the shared group state over time.
- Messages sent between members of the group (Application messages).

Marmot add to group example

Alice wants to be included by Bob in a group.

1. Alice publishes a **KeyPackage Relays List Event** (where to find her 🎫 KeyPackage)
2. Alice publishes a 🎫 **KeyPackage event** (her invitation credentials)
3. Bob gets Alice's 🎫 **KeyPackage**, and uses it to add her to the group (via an MLS 'commit message' 💬)
4. Bob sends Alice a 👋 **Welcome Event** via Gift Wrap 🎁
5. Alice processes the 👋 **Welcome Event** and can start messaging 💬

What exists today

Marmot Protocol

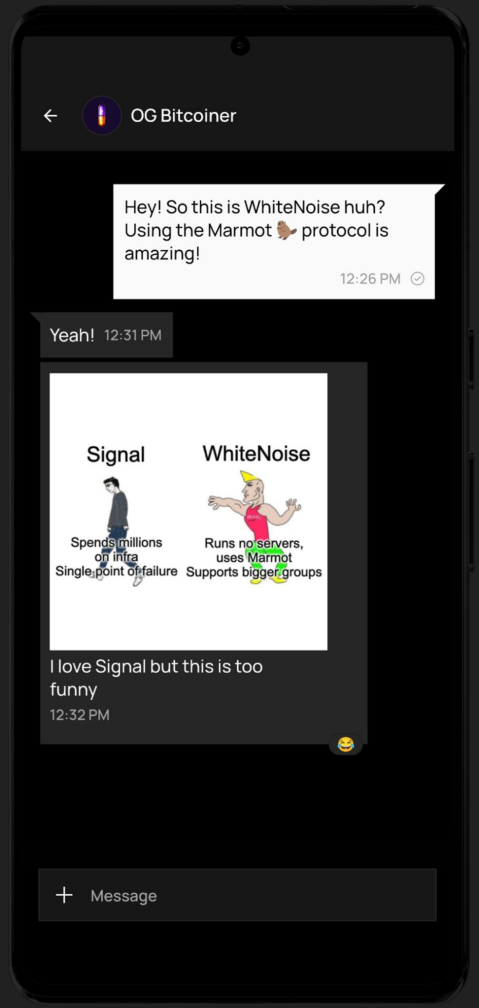
Protocol Specs https://github.com/parres-hq/marmot

MDK - Marmot Development Kit

Open source rust implementation of the Marmot protocol

WhiteNoise

Secure, private, and decentralized chat app built using MDK



The Vision

No company to shut down

No servers to seize

No single point of failure

Available today

Open source

Anyone can build on it

Freedom tech that actually works

Key Takeaways

Privacy needs censorship resistance

Group messaging needs scalability

Marmot delivers both

Get Involved

📖 Read the Marmot specs and give us feedback!

https://github.com/parres-hq/marmot

🔧 Build with MDK

https://github.com/parres-hq/mdk

📱 Try WhiteNoise

https://whitenoise.chat

Extra Resources

- NIP-04 considered harmful https://github.com/nostr-protocol/nips/issues/107
- NIP-17 https://github.com/nostr-protocol/nips/blob/master/17.md
- NIP-44 (Encrypted Payloads) https://github.com/nostr-protocol/nips/blob/master/44.md
- NIP-59 (Gift Wrap) https://github.com/nostr-protocol/nips/blob/master/59.md
- NIP-EE https://github.com/nostr-protocol/nips/blob/master/EE.md
- Marmot https://github.com/parres-hq/marmot
- MLS RFC 9420 https://www.rfc-editor.org/rfc/rfc9420.html