Security and Privacy FAQ

- Is Meshtastic encrypted? Can anyone read my messages?
- Who can see my location on the mesh?
- Can I trust MeshCore encryption for sensitive communications?

Is Meshtastic encrypted? Can anyone read my messages?

Short Answer

Meshtastic messages are encrypted, but the level of protection depends on which channel you're using. The default channel (LongFast) uses a known public key and provides essentially no privacy. Custom channels with randomly generated keys provide strong message confidentiality.

The Default Channel: Not Private

The default Meshtastic channel uses a Pre-Shared Key (PSK) of `AQ==`, which is base64 for a single byte of value `0x01` - a publicly known and documented default key. Any Meshtastic user in radio range can read messages on the default channel, including the Meshtastic app developers and anyone who has read the public documentation.

The default channel is suitable for public community communication where privacy isn't a concern. Do not send anything private on the default channel.

Custom Channels: Strong Privacy

When you create a channel with a randomly generated PSK (using the app's random key generator), the app can produce a 256-bit AES key that cannot be recovered by brute force with current technology. Note that Meshtastic uses AES-128 or AES-256 in CTR mode depending on the key length (16 vs 32 bytes) - the default and many shared channels are AES-128, while the app's random-key generator can produce a 256-bit key. Messages on this channel are readable only by nodes that have the same PSK.

Encryption security: AES-256 is a strong, government-approved cipher and the cryptography itself is sound. However, security depends on key management (how you distribute the PSK to your community), and Meshtastic exposes metadata and relies on manual key distribution - so do not treat a custom channel as government-grade secure for highly sensitive traffic.

Direct Messages: Even Better (Firmware 2.5+)

Direct messages use X25519 (Curve25519) ECDH public-key (PKI) encryption. PKI direct messaging was introduced in firmware 2.3 and became the robust default in 2.5; run firmware 2.5 or later on all nodes for current PKI DM security. It provides:

- End-to-end encryption between just sender and recipient
- No shared secret to distribute - keys are derived automatically from each node's public/private key pair

Note: Meshtastic does **not** provide forward secrecy. Because the keys are long-lived, traffic captured today can be decrypted later if a key is compromised - the official documentation notes that Meshtastic is vulnerable to "harvest now, decrypt later" attacks. Do not assume that seizing a key in the future cannot expose past messages.

What an Eavesdropper Can See

Even with properly configured channel encryption, a radio observer can see:

- That LoRa transmissions are occurring on the frequency
- The approximate timing and frequency of transmissions
- Some packet header fields that are not encrypted (node IDs, a channel hash, hop count, and routing/metadata)

Message content, sender names, and channel names are inside the encrypted portion of the packet, so they are hidden if a custom PSK is in use. However, the unencrypted header still exposes node IDs, a channel-hash byte, and routing/metadata - so an eavesdropper can still learn who is transmitting and when.

Practical Recommendations

- For community chat where privacy isn't critical: default channel is fine

- For any sensitive coordination: create a custom channel with a random PSK
- For private one-on-one messages: use DMs with firmware 2.5+ on both ends
- For highly sensitive communications: LoRa mesh is supplemental - use Signal or other end-to-end encrypted messaging for truly sensitive content

Who can see my location on the mesh?

How Position Data Spreads

When position reporting is enabled on a Meshtastic node, GPS coordinates are broadcast as position packets on the mesh. These packets travel through the network like any other message and are visible to all nodes that receive them.

Who Can See Your Position

- **Anyone on your channel** - If you're on the default channel, anyone with Meshtastic within a few hops can see your position in their node list and on their map
- **meshmap.net** - If you have MQTT uplink enabled to the public Meshtastic broker, your position appears on the global map
- **MQTT operators** - Anyone subscribed to the same MQTT topic receives your position packets

Controlling Position Privacy

Option 1: Disable Position Reporting

Turn off GPS or disable position broadcasts entirely:

```
meshtastic --set position.gps_mode DISABLED
meshtastic --set position.position_broadcast_secs 0
```

Your node will still relay messages and appear in node lists (by node ID), but without location data. Note that disabling position only hides your GPS coordinates: the node still transmits its node ID and (by default) its name, and the transmitter itself can be located by radio direction-finding. For OPSEC-sensitive field or shelter operations, also anonymize the node name, minimize

transmissions, and remember that the mere presence of a radio signal is detectable.

Option 2: Use Position Precision Reduction

Broadcast a less precise location rather than exact GPS coordinates:

```
meshtastic --set position.position_precision 10
```

The `position_precision` value sets how many bits of latitude/longitude are shared, which maps to a much coarser ground radius than the raw number suggests. Approximate values (per the Meshtastic precision_bits reference): 32 = full precision; 19 ≈ ±46 m; 16 ≈ ±365 m; 13 ≈ ±2.9 km; 11 ≈ ±11.7 km; 10 ≈ ±23 km; 0 = position never sent. Note these are coarser than they appear - for example, precision 16 still exposes a roughly 365 m radius, not 1 km, so choose conservatively. With precision 10 (±23 km) you appear somewhere in a wide area around your city rather than at your exact street address.

Option 3: Private Channel with No MQTT

Create a private channel and disable MQTT uplink. A private channel with MQTT disabled keeps your position content out of the encrypted payload that reaches the public maps, so your coordinates don't appear there. It is not total invisibility, though: position packets still travel over RF, and unencrypted header fields (node IDs, hop data) remain observable to any radio listener. Privacy also depends on every node on the channel keeping MQTT/uplink disabled - if any participant later enables MQTT or bridges to the internet, position can leak.

Fixed Position vs GPS

For fixed infrastructure nodes (repeaters), configure a static position manually rather than using GPS. This gives you explicit control over what coordinates you publish, and you can choose a slightly offset position if you prefer not to reveal your exact rooftop address.

Can I trust MeshCore encryption for sensitive communications?

MeshCore Encryption Summary

MeshCore provides two layers of encryption:

- **Channel encryption** - AES-128 ECB + HMAC-SHA256 with a key derived from the channel configuration. All nodes on the channel share this key. Note that public (no-PSK) channels and open hashtag rooms are readable by any passive listener.
- **Direct message encryption** - ECDH key exchange (Curve25519 elliptic curve) with AES session keys. Only sender and recipient can read DMs.

Cryptographic Strength

Both AES-128 and Curve25519 ECDH are modern, vetted cryptographic primitives used in TLS 1.3, Signal Protocol, and other high-security applications. Note, however, that those protocols use authenticated (AEAD) modes, whereas MeshCore's mode of use - AES-128 in ECB mode with a short 2-byte authentication tag and static ECDH keys - is weaker than the AEAD constructions in TLS 1.3 and Signal. MeshCore direct messages use sound primitives (X25519 ECDH key agreement + AES-128) but in a much simpler construction than Signal. Specifically, it uses AES-128 in ECB mode, a short 2-byte authentication tag, and static (long-term) keys - so it does NOT provide forward secrecy or the ratcheting that Signal uses. Treat it as basic confidentiality against casual interception, not as equivalent to Signal for high-sensitivity traffic.

What Encryption Protects Against

- **Passive eavesdroppers with LoRa hardware** - Cannot read message content with correct key management. (Public, no-PSK channels and open hashtag rooms remain

readable by passive listeners.)
- **Casual interception of DM content** - Without the key, an eavesdropper cannot read DM plaintext. Note, however, that because DMs use AES-128 in ECB mode, identical plaintext blocks produce identical ciphertext, so an observer recording traffic long-term may detect repeated or structured content patterns (for example, templated emergency-comms messages) even without the key. Avoid sending fixed-format sensitive messages.
- **Replay/recorded-traffic decryption** - MeshCore's DM ECDH uses static long-term identity keys, not ephemeral session keys, so it does NOT provide forward secrecy. If a device's private key is later recovered, previously recorded DMs to or from that device can be decrypted. Replay resistance, where present, comes from per-message timestamps, not from the key exchange.

What Encryption Does NOT Protect Against

- **Physical access to your device** - Private keys are stored on the device. Seizure of the device potentially allows recovery of stored messages.
- **Compromised network operator** - A room server operator can see metadata (who is communicating with whom, when) and the content of room/channel posts that pass through the server; only person-to-person DMs are end-to-end encrypted and not readable by the operator.
- **Traffic analysis** - Observers can see that radio transmissions are occurring, timing patterns, and frequency of communication even if they cannot read content
- **Social engineering** - If a recipient shares a DM, encryption provides no protection
- **Channel key compromise** - If the shared channel key is obtained by an adversary, all past channel traffic (if recorded) can be decrypted
- **Loss of forward secrecy** - Because DM keys are static, recovering a private key compromises past as well as future messages; there is no per-message ratcheting like Signal's Double Ratchet.

Appropriate Use Cases

MeshCore's encryption is appropriate for:

- Community coordination that you'd prefer not to broadcast publicly
- Emergency operations where commercial networks are unavailable - but only as a supplemental, best-effort text channel. Mesh has no delivery guarantee, low throughput, and degrades under congestion; maintain a primary emergency-comms method and never rely on mesh alone for life-safety traffic. The encryption above provides basic confidentiality only and is not suitable for life-safety or classified-equivalent secrecy.

- Private group communications within a trusted community

MeshCore encryption is *not* appropriate as the sole protection for:

- Communications subject to legal privilege (attorney-client, medical)
- Operational security against nation-state adversaries
- Highly sensitive personal information

For these use cases, use end-to-end encrypted applications (Signal, ProtonMail) with LoRa mesh serving only as a transport layer to reach an internet gateway.