Encryption and Privacy

- PKC Direct Messaging (v2.5+)
- Meshtastic Managed Mode and Admin Channels

PKC Direct Messaging (v2.5+)

Meshtastic v2.5 introduced **Public Key Cryptography (PKC) encrypted direct messages** - a significant security upgrade that encrypts DM content per-recipient rather than relying only on the shared channel key. Note that PKC protects the message *body*: the packet header (source/destination node IDs, hop count, timing) remains plaintext and can be uplinked to MQTT, so PKC protects what you say, not the fact that two nodes are communicating.

Note on terminology: This feature is officially called "PKC Direct Messages" or "encrypted direct messages" in Meshtastic documentation. PKC encrypted direct messages were introduced in **firmware v2.5.0** (not v2.3 or v2.4 as some sources incorrectly state).

Before v2.5: How DMs Worked

Prior to v2.5, "direct messages" in Meshtastic were standard channel messages with a `to` field set to the recipient's node ID. Anyone on the same channel with the channel key could decrypt and read all DMs. There was no per-recipient encryption.

v2.5+: PKC Encrypted Direct Messages

From v2.5 onward, direct messages use per-node asymmetric encryption:

- **Key exchange: X25519 ECDH** - each node has an X25519 public/private key pair
- **Encryption: AES-CCM** - using the derived shared secret as the key (per the Meshtastic encryption documentation)
- Only the intended recipient can decrypt the message - the channel key is not used
- Node public keys are distributed automatically via NodeInfo packets

Limitations: PKC has **no forward secrecy** — keys are long-lived, so the "harvest now, decrypt later" risk applies if a node's private key is ever compromised — and there is **no key revocation** mechanism. The plaintext packet header still leaks metadata even on PKC DMs.

Backward Compatibility

If you send a PKC-encrypted DM to a node running firmware 2.4.3 or older, Meshtastic automatically falls back to the legacy channel-based method (which is unprotected against anyone holding the channel key). Some app versions display a key/lock icon indicating that PKC was used for a given message.

Requirements

- Both sender and recipient must be running Meshtastic firmware **v2.5 or later**
- Both nodes must have exchanged NodeInfo packets (public keys are included automatically)
- Compatible with Android, iOS, and Python CLI clients that support v2.5+

Source: meshtastic.org/docs/overview/encryption/ and meshtastic.org/blog/introducing-new-public-key-cryptography-in-v2_5/. Verified 2026-05-03.

Meshtastic Managed Mode and Admin Channels

For deployed infrastructure nodes - community repeaters, fixed gateways - you want to prevent unauthorized configuration changes while still being able to administer the node remotely. Meshtastic provides two tools for this: Managed Mode and Admin Channels.

Managed Mode

When Managed Mode is enabled, the node blocks all client applications from *writing* configuration over any local interface - BLE, USB serial, and TCP alike (configuration can still be read). Once enabled, configuration can only be changed through PKC Remote Admin (firmware 2.5+) or the legacy Admin channel (firmware prior to 2.5). This prevents anyone who walks up to the repeater and pairs their phone from changing the configuration.

Managed Mode is set via the `security.is_managed` boolean, **not** a device role - there is no "MANAGED" device role:

```
meshtastic --set security.is_managed true
```

Configure and test your admin access (admin channel or PKC remote admin key) **before** enabling `security.is_managed`. If you enable Managed Mode without working admin access, you lose the ability to reconfigure the node over BLE, USB serial, and TCP - the only recovery is a physical firmware erase / factory reset, which also wipes your keys and configuration.

With Managed Mode active:

- Local config writes over BLE, USB serial, and TCP are all blocked (changes require Remote Admin or the admin channel)
- Configuration can still be read over local interfaces
- The node continues to route and relay normally

Admin Channel

The Admin Channel is an encrypted control channel that allows authorized administrators to configure any node in the mesh remotely - even nodes that are out of direct radio range (configuration packets are relayed through the mesh). On firmware 2.5 and newer, the preferred approach is PKC Remote Admin, which authorizes administrators by their public key rather than a shared channel PSK; the legacy shared-PSK admin channel remains for older firmware.

Setting Up an Admin Channel

1. Create a channel with a random PSK and name it "admin" (or any name you choose)
2. Designate that channel as the admin channel (the security admin-channel setting)
3. Add this channel to all nodes you want to manage
4. Only administrators should have the admin channel PSK

```
meshtastic --ch-set name "admin" --ch-index 1
meshtastic --ch-set psk random --ch-index 1
meshtastic --set security.admin_channel_enabled true
```

The legacy integer `admin_channel_index` field is deprecated; on current firmware the admin channel is enabled via the `security.admin_channel_enabled` boolean. On firmware 2.5+, prefer PKC Remote Admin (admin keys) over the shared-PSK admin channel where possible.

Remote Configuration via Admin Channel

Once an admin channel is configured, you can send configuration commands to remote nodes via the app's remote admin feature. The command is encrypted with the admin channel PSK (or, with PKC remote admin, the admin keypair), relayed through the mesh, and executed on the target node. The target node responds with its updated configuration.

Security Considerations

- The admin channel PSK (or PKC admin key) is the master key for your infrastructure - anyone holding it can administer every node that trusts it, so guard it carefully. Managed Mode does *not* protect against an attacker who already has this credential.
- Distribute admin credentials only to trusted operators
- Consider a separate admin channel/key per node, or per geographic cluster, to limit blast radius if a key is compromised
- Managed Mode blocks normal config writes over USB serial too. Physical access still allows a firmware erase / factory reset, which wipes Managed Mode, keys, and configuration - but that is a reset, not live reconfiguration of a running node.