

Monero: Private and Decentralized Cryptocurrency

This chapter covers the setup, management, and remote access of a self-hosted Monero node and wallet. It includes detailed guides on daemon configuration, wallet CLI usage, Tailscale integration for secure remote connections, troubleshooting, and best practices for maintaining privacy and security within the Monero ecosystem.

- [Monero: Privacy by Design – Origins, Technology, and Ongoing Evolution](#)
- [Monero Monitoring & Troubleshooting Commands Reference](#)
- [Monero Wallet Guide](#)
- [Secure Remote Access to Monero Node Using Tailscale VPN](#)
- [Monero Node Administration & Maintenance Guide](#)
- [🔧 Allowing Incoming Connections to Monerod \(Port 18080\) on DietPi](#)
- [🔧 Creating and Managing Additional Accounts for Transaction Segmentation in Monero Wallet](#)

Monero: Privacy by Design – Origins, Technology, and Ongoing Evolution

Overview

[Monero](#) is a privacy-focused, decentralized cryptocurrency built to deliver strong financial confidentiality by default. Unlike Bitcoin and other cryptocurrencies that offer optional or partial privacy features, Monero was designed from the ground up to be untraceable and anonymous. It achieves this through the use of advanced cryptographic techniques, including ring signatures, stealth addresses, confidential transactions, and bulletproofs.

At its core, Monero values decentralization and user sovereignty. To maintain a level playing field for miners and avoid hardware-based centralization, the project regularly changes its proof-of-work (PoW) algorithm to resist ASIC dominance.

Privacy Technologies

- **Ring Signatures:** Obscure the origin of a transaction by combining the sender's output with decoy outputs from other users, making it computationally infeasible to determine the actual source.
 - **Stealth Addresses:** Each transaction generates a unique, one-time address for the recipient, preventing transactions from being linked to their wallet address.
 - **Ring Confidential Transactions (RingCT):** Hide the transaction amount in addition to sender and recipient data. Introduced in 2017 and based on a proposal by Bitcoin Core developer Greg Maxwell, RingCT significantly enhances overall privacy.
 - **Bulletproofs:** Implemented in 2018, this zero-knowledge proof technology replaced the older range proofs used in confidential transactions. Bulletproofs reduce transaction size by over 80%, improving scalability and efficiency.
 - **Fungibility:** Since Monero coins have no visible transaction history, they are inherently fungible—each coin is indistinguishable from another, unlike Bitcoin, where coins can be “tainted” based on past activity.
-

Origins and History

Monero began as a fork of **Bytecoin**, the first implementation of the **CryptoNote** protocol—a system designed to address privacy issues, mining centralization, and uneven coin distribution in Bitcoin. Bytecoin launched in March 2014 but was marred by controversy over an 80% premine.

In response, a Bitcointalk user known as **thankful_for_today** forked the Bytecoin code into a new project called **BitMonero**—a fusion of "Bit" (Bitcoin) and *Monero* (Esperanto for "coin"). The launch of BitMonero was poorly received, prompting seven community members to fork it again under the simplified name **Monero**. This group, led by the pseudonymous developer **Fluffypony** (Riccardo Spagni), became the first Monero Core team.

From its inception in April 2014, Monero has operated without a premine or founder rewards, aligning with its ethos of fairness and decentralization.

Monero's strong privacy features soon made it a favored cryptocurrency among users seeking financial confidentiality. It also became one of the most used currencies on darknet markets—a reality that has contributed to both its popularity and controversy.

Ongoing Development

Monero evolves through **scheduled hard forks**, typically occurring every six months. These upgrades introduce protocol improvements, fix bugs, and often include changes to the PoW algorithm to reinforce decentralization.

Notable milestones include:

- **2017:** Introduction of RingCT, enabling complete transaction privacy.
 - **2018:** Adoption of bulletproofs, reducing transaction size and cost.
 - **2019:** Resignation of lead maintainer Fluffypony, a move intended to further decentralize project governance.
-

Regulatory and Social Context

Monero's unwavering focus on anonymity has drawn mixed reactions. Supporters champion it as a vital tool for privacy rights and financial autonomy. Critics, including regulators, express concern about its potential misuse in illicit activity.

Some jurisdictions have considered or enacted restrictions on Monero due to its untraceable nature. Despite this, Monero remains a legitimate cryptocurrency with a dedicated global community and ongoing technical innovation.

Conclusion

Monero stands out in the cryptocurrency space for its strong commitment to privacy, decentralization, and fungibility. While its anonymity features spark ongoing debate, Monero continues to push forward as a leading project for those who believe financial privacy is not just a feature—but a fundamental right.

See also:

- [Monero Wallet Guide](#)
- [Monero Monitoring & Troubleshooting Commands Reference](#)

Monero Monitoring & Troubleshooting Commands Reference

1. Check Monero Daemon Status

```
systemctl status monerod
```

2. Start / Stop / Restart Monero Daemon

```
systemctl start monerod  
systemctl stop monerod  
systemctl restart monerod
```

3. Check Real-Time Syncing Progress (Logs)

```
journalctl -u monerod.service -n 20 --no-pager
```

Or continuously (live):

```
tail -f /mnt/monero/.bitmonero/monerod.log
```

4. Check if Monero is Running

```
ps aux | grep monerod
```

5. List All Screen Sessions

```
screen -ls
```

6. Reattach to Existing Screen Session

```
screen -r <session_name>
# e.g., screen -r monero
```

7. Detach from Screen (keep running in background)

```
Ctrl + A then D
```

8. Check Disk Space

```
df -h
```

Specific mount point:

```
df -h /mnt/monero
```

9. Inspect Large Files/Folders

```
du -xh / | grep -P 'G\t' | sort -hr | head -20
```

10. Check for Deleted Files Still Using Disk Space

```
lsdf | grep deleted
```

11. Check Wallet Sync Status from CLI

Inside `monero-wallet-cli`:

```
status
net_stats
```

12. Manually Launch Wallet CLI

```
/mnt/monero/monero-wallet-cli --wallet-file /mnt/monero/wallets/blueace
```

13. Account Tag and Label Commands

Inside wallet CLI:

```
account tag <tag_name> <index>
account tag_description <tag_name> <description>
account label <index> <label text>
```

14. Ensure Mount on Boot

In `/etc/fstab`, add:

```
UUID=<your-uuid> /mnt/monero ext4 defaults 0 2
```

Then:

```
mount -a
```

15. Quick Ping to Confirm Internet/DNS

```
ping -c 3 deb.debian.org
```

This set of commands should cover most operational and recovery scenarios. Feel free to expand this as your setup evolves (e.g., adding wallet RPC, remote access, VPN, or port monitoring).

See also:

- [Monero Wallet Guide](#)
- [Monero Privacy by Design - Origins, Technology, and Ongoing Evolution](#)

Monero Wallet Guide

? Create a New Wallet

```
/mnt/monero/monero-wallet-cli --generate-new-wallet /mnt/monero/wallets/<wallet_name>
```

- Replace `<wallet_name>` with your desired wallet filename.
- You will be prompted to set a password and make a secure note of your seed phrase.

? Open an Existing Wallet

```
/mnt/monero/monero-wallet-cli --wallet-file /mnt/monero/wallets/<wallet_name>
```

- Replace `<wallet_name>` with the name of the wallet you want to open (e.g., `nero`).

? Mount Wallet Directory (if needed)

If your wallet directory is not mounted automatically on boot:

```
mount /dev/sda1 /mnt/monero
```

To mount it automatically on boot, add the following line to `/etc/fstab`:

```
/dev/sda1 /mnt/monero ext4 defaults,noatime 0 2
```

?? Address Labeling

Create a new subaddress with a label:

```
address new <label>
```

- Example: `address new exchange_withdrawal`

View all addresses and their labels:

```
address all
```

Change an existing label:


```
address label <index> <new label>
```

- Index is the number listed with `address all` (e.g., `1`).

? Tag Management

Assign a tag to a specific account index:

```
account tag <tag_name> <account_index>
```

- Example: `account tag personal 0`

Set a description for a tag:

```
account tag_description <tag_name> <description>
```

- Example: `account tag_description personal "casual spending"`

? Copy Wallet Files to Your Local Machine (via SCP)

From your **local machine**, run:

```
scp root@<remote_ip>:/mnt/monero/wallets/<wallet_name>* ~/Downloads/
```

```
scp 'root@192.168.1.155:/mnt/monero/wallets/nero*' ~/Downloads/ # on Mac
```

- Replace `<remote_ip>` with your Pi's IP address (e.g., `192.168.1.155`).
- Replace `<wallet_name>` with your actual wallet name (e.g., `blueace`).

? How to Use Wildcards with `scp` on macOS (zsh)

When using the `scp` command in the **macOS Terminal**, you might run into this error:

```
zsh: no matches found: root@192.168.1.155:/mnt/monero/wallets/nero*
```

This happens because **zsh**, the default shell on macOS, tries to expand the `*` **wildcard locally on your Mac** before sending the command to the remote machine. Since there's

no local file named `nero*`, it throws an error.

? Solution

To prevent zsh from expanding the wildcard, **quote the remote path** in your `scp` command:

```
scp 'root@192.168.1.155:/mnt/monero/wallets/nero*' ~/Downloads/ # copy Monero wallet to Mac
```

“`[]` Always use **single quotes** `' '` to preserve the literal meaning of wildcards for the remote shell.

Let me know if you'd like this saved in a reusable note or command reference

? Why Avoid Using the Primary Address

- **Primary address reuse reduces privacy** by making it easier to associate transactions.
- **Best practice:** Always use a new subaddress per transaction or recipient.
- Monero's built-in subaddress system enhances unlinkability and protects financial metadata.

? Sweep Balance to Another Wallet

To transfer all funds to another address:

```
sweep_all <destination_address>
```

- Used to migrate to a new wallet like `nero` from `blueace`.

See also:

- [Monero Privacy by Design - Origins, Technology, and Ongoing Evolution](#)
- [Monero Monitoring & Troubleshooting Commands Reference](#)

Secure Remote Access to Monero Node Using Tailscale VPN

Q: What VPN solution are we using to securely access the Monero node remotely?

A: We use **Tailscale**, a user-friendly mesh VPN based on WireGuard, to create a secure, private network between your devices.

Q: How do I install Tailscale on my DietPi (Monero) server?

A: Run this command on the DietPi terminal to install Tailscale:

```
curl -fsSL https://tailscale.com/install.sh | sh
```

Q: How do I start and enable Tailscale service on DietPi?

A: After installation, start the service and enable it to run at boot:

```
sudo systemctl enable --now tailscaled
```

Q: How to authenticate and connect the DietPi server to your Tailscale network?

A: Run this to authenticate:

```
sudo tailscale up
```

Q: How do I check that my DietPi server is connected to the Tailscale network?

A: Use the status command:

```
tailscale status
```

You should see your DietPi listed with its Tailscale IP address.

Q: How to install Tailscale on your local Mac (or other client machine)?

A: Download the Tailscale app from [tailscale](https://tailscale.com) and install it. Then sign in with the same Tailscale account you used for the server.

Q: How do I confirm the local machine is connected and can reach the DietPi server?

A: On your Mac terminal, ping the DietPi Tailscale IP:

```
ping <dietpi-tailscale-ip>
```

Replace `<dietpi-tailscale-ip>` with the actual Tailscale IP you see in `tailscale status`.

Q: How do I connect my Monero wallet CLI on my Mac to the remote monerod daemon over Tailscale?

A: Use the remote daemon IP and port in your wallet CLI command:

```
./monero-wallet-cli --wallet-file /path/to/yourwallet --daemon-address <dietpi-tailscale-ip>:18081 --trusted-daemon
```

Q: What if I encounter connection errors or timeouts?

A: Verify:

- The monerod daemon is running and listening on `0.0.0.0:18081` on DietPi.
 - Firewall rules on DietPi allow incoming connections on port 18081.
 - Tailscale network shows both devices connected.
 - You can ping and `nc` (netcat) test the port from your local machine.
-

Q: How to stop or disable Tailscale on DietPi if needed?

A:

```
sudo systemctl stop tailscaled
```

```
sudo systemctl disable tailscaled
```

Monero Node Administration & Maintenance Guide

1. Checking for Duplicate Files and File Usage

- **Find duplicate files by name** (search for specific file names):
`find / -type f \(-name "output0.csv" -o -name "output1.csv" \) 2>/dev/null`
- **Check file disk usage** (with depth level, e.g., 1 for directory contents):
`du -h --max-depth=1 /path/to/dir`

2. Open Ports and What Services They Listen To

- **Check for open ports and the services they listen to** (view listening ports and active connections):
`netstat -tuln`
- **Find which services are using specific ports (e.g., port 18080):**
`lsof -i :18080`
- **Check if specific ports are open (e.g., 18080 or 22):**
`netstat -tuln | grep 18080`
- **List open ports with `ss` (useful alternative to `netstat`):**
`ss -tuln`

3. Checking iptables and UFW Rules

iptables Commands:

- **View iptables rules:**
`iptables -L`
- **View specific chain rules (e.g., INPUT chain):**
`iptables -L INPUT -n --line-numbers`

- **Show a specific port's status in iptables** (e.g., port 22 or 18080):
`iptables -L INPUT -n --line-numbers | grep -E '22|18080'`
- **View specific rules in a named chain** (e.g., `ts-input`):
`iptables -L ts-input -n --line-numbers`
- **Delete an iptables rule** (e.g., delete rule number 3 in the INPUT chain):
`iptables -D INPUT 3`
- **List all iptables rules including the ones set by UFW:**
`iptables -L`
- **ufw (Uncomplicated Firewall):**
`ufw status`
- **Enable UFW:**
`ufw enable`
- **Allow incoming traffic on port 18080** (for Monero node):
`ufw allow 18080/tcp`
- **Check UFW status and rules:**
`ufw status verbose`
- **Disable UFW** (if necessary):
`ufw disable`

4. Network and Firewall Troubleshooting

- **Check IP routing or address information:**
`ip a`
- **Check active connections and the state of the ports:**
`ss-tuln`

5. Commands for Managing Services and Systemd

- **Check the status of a service** (e.g., `monerod`):
`systemctl status monerod`

- **Stop a service (e.g., monerod):**
`systemctl stop monerod`
- **Start a service (e.g., monerod):**
`systemctl start monerod`
- **Enable a service to start on boot (e.g., monerod):**
`systemctl enable monerod`
- **Disable a service from starting on boot:**
`systemctl disable monerod`

6. Mail Setup (msmtp, Mailutils)

- **Send a test email using msmtp:**
`echo "Test email" | msmtp --debug user@example.com`
- **Check mail logs for issues** (e.g., for msmtp sending failures):
`cat /var/log/mail.log`
- **Configure msmtp for sending emails (via SMTP):** Edit the `~/.msmtprc` file to include the email credentials and SMTP server details.

7. Directory and File Clean-Up

- **Remove directories and files** (e.g., remove `.bitmonero`):
`rm -rf /path/to/directory`
- **Delete unnecessary files (e.g., log files):**
`rm /path/to/file.log`

8. System Information and Disk Usage

- **Show system disk usage:**
`df -h`
- **View detailed file sizes in directories:**
`du -sh /path/to/directory`
- **Show the current system resource usage:**
`top`

9. Miscellaneous Useful Commands

- **Check the current kernel version:**
`uname -r`
- **Search for a specific string in files:**
`grep "search_string" /path/to/file`
- **View the contents of a file:**
`cat /path/to/file`

10. File Transfer Commands

- **Transfer files via SCP** (e.g., from your Pi to local):
`scp root@your_pi_ip:/path/to/file /local/destination`
- **Transfer files via SCP with wildcards:**
`scp root@your_pi_ip:/path/to/files/* /local/destination/`
- **Transfer files via SCP from Pi to Mac** (Mac uses a similar approach to Linux for SCP):
`scp 'root@your_pi_ip:/path/to/file' ~/local/destination`
Example:
`scp 'root@192.168.1.155:/mnt/monero/wallets/nero*' ~/Downloads/`

? Allowing Incoming Connections to Monerod (Port 18080) on DietPi

A guide to enabling and managing `iptables` rules to support incoming P2P traffic on TCP port `18080`, used by the Monero daemon (`monerod`).

? Current Setup Summary

- **System:** DietPi (Debian-based, minimal)
- **Firewall:** `iptables` with a `ts-input` chain (Tailscale or custom)
- **Monero Daemon:** `monerod` running with `--p2p-bind-ip=0.0.0.0`
- **Problem:** No incoming P2P connections; Monero logs show:

```
W No incoming connections - check firewalls/routers allow port 18080
```

? Allow TCP Port 18080 (Temporary Rule)

Use this command to add the rule immediately (effective until next reboot):

```
iptables -I ts-input 1 -p tcp --dport 18080 -j ACCEPT
```

Check that the rule was added:

```
iptables -L ts-input -n --line-numbers | grep 18080
```

? Confirm monerod Is Listening

Check if your node is listening on the correct port:

```
ss -tnl | grep 18080
```

Or:

```
lsof -i :18080
```

Expected result:

```
monerod ... TCP *:18080 (LISTEN)
```

? Save Rule (Non-Persistent Until Fixed)

Manually save the current iptables rules:

```
iptables-save > /etc/iptables/rules.v4
```

Verify it's in the file:

```
grep 18080 /etc/iptables/rules.v4
```

Expected result:

```
-A ts-input -p tcp -m tcp --dport 18080 -j ACCEPT
```

? Persistence Issue (DietPi)

Although `iptables-persistent` was installed and `rules.v4` was written correctly, the rule **did not survive reboot**. A systemd workaround was attempted but failed.

? Clean-Up: Remove systemd Workaround

If used earlier, remove the failing custom service:

```
systemctl disable restore-iptables.service rm /etc/systemd/system/restore-iptables.service  
systemctl daemon-reload
```

? Practical Workaround

Until persistence is resolved, run this command manually after each reboot:

```
iptables -I ts-input 1 -p tcp --dport 18080 -j ACCEPT
```

Optional helper script:

```
echo "iptables -I ts-input 1 -p tcp --dport 18080 -j ACCEPT" > ~/fix-monero-fw.sh chmod +x ~/fix-monero-fw.sh
```

? View Live Monero Logs

Monitor `monerod` output in real-time:

```
journalctl -u monerod -f
```

? Intermittent “No Incoming Connections” Warnings Are Normal

It's common for `monerod` to log messages like:

```
W No incoming connections - check firewalls/routers allow port 18080
```

This warning appears **once per hour** if your node has had **zero inbound peer connections** in the previous hour.

As long as:

- `monerod` is listening on `0.0.0.0:18080`
- The correct `iptables` rule is applied
- Port 18080 is forwarded in your router
- You see outbound `ESTABLISHED` connections in `lsof` or `ss`

...then your node is healthy and participating in the network.

Inbound peers will connect intermittently, especially after a reboot or on newer nodes. Once a peer connects to your node, these warnings will stop until the inbound count returns to zero again.

? Final Status

- Inbound P2P connections to Monero now work **after rule is applied**

- Warning in Monero logs no longer appears
- Node now supports the network as a full peer
- Persistence can be revisited later using `nftables`, `rc.local`, or cron

? Creating and Managing Additional Accounts for Transaction Segmentation in Monero Wallet

? Create a New Account

```
account new <label>
```

- Example: `account new donations`

?? Create and Label a Subaddress

```
address new <label>
```

- Example: `address new Dev_Team`

? Assign a Tag to an Account

```
account tag <tag_name> <account_index>
```

- Example: `account tag donate 1`

? Set a Tag Description

```
account tag_description <tag_name> <description>
```

- Example: `account tag_description donate "Donations to causes and projects"`

? Send Funds from Account 0 to Account 1

First, switch to account 0 (if not already selected):

```
account switch 0
```

Then send funds to the Dev Team subaddress in account 1:

```
transfer <destination_address> <amount>
```

- Example:

```
transfer 88xkUv...xyz 0.04
```

? Notes on Monero Account and Address Behaviour

- Monero **treats each account holistically**. Even if funds are sent from a subaddress, the account's **primary address** may become marked as "used" after a transaction.
- This does not compromise privacy — it simply reflects internal wallet indexing.
- Using **subaddresses** for sending/receiving helps keep your financial metadata compartmentalised.
- **New accounts** can be used to segregate funds for donations, savings, or specific activities.

?? Address Labelling

Create a new subaddress with a label:

```
address new <label>
```

- Example: `address new exchange_withdrawal`

View all addresses and their labels:

```
address all
```

Change an existing label:

```
address label <index> <new label>
```

- Index is the number listed with `address all` (e.g., `1`).

? Tag Management

Assign a tag to a specific account index:

```
account tag <tag_name> <account_index>
```

- Example: `account tag personal 0`

Set a description for a tag:

```
account tag_description <tag_name> <description>
```

- Example: `account tag_description personal "casual spending"`
-