# LLM Remote

Encrypted Telegram ↔ AI Multi-Provider Bridge

REDEGAL · DIGITAL CONSULTING GROUP

Version 1.3 · February 2026 · Internal Document

## Table of Contents



# 1. Introduction

**LLM Remote** is an internal tool that allows Redegal team members to interact with multiple AI providers (Claude Code, OpenAI, Gemini, Groq) directly from Telegram, with bank-grade encryption.

## What is it for?

- **Remote development** — Run Claude Code on your machine from anywhere
- **Quick queries** — Ask GPT-4o, Gemini or Llama from your phone
- **Multi-project** — Switch between projects with a single command
- **Secure** — Everything encrypted, authenticated, and audit-logged

> **Prerequisites**
>
> Node.js 20+, a Telegram bot (from @BotFather), and optionally Claude Code CLI.

# 2. AI Providers

LLM Remote supports 5 providers. Switch between them with `/ia <name>` in Telegram.

| Provider | Command | Mode | Cost |
|---|---|---|---|
| 🟣 **Claude Code** | `/ia claude` | **Agentic** Reads/writes files, runs commands | **Plan-based** |
| 🟢 **OpenAI GPT-4o** | `/ia openai` | Chat API | **Pay-per-use** |
| 🔵 **Gemini 2.5 Flash** | `/ia gemini` | Chat API | **Free** 20 req/day |
| 🟠 **Groq Llama 3.3** | `/ia groq` | Chat API (ultra-fast) | **Free** 30 req/min |
| 🟣 **Anthropic Sonnet** | `/ia anthropic` | Chat API (non-agentic) | **Pay-per-use** |

> **Daily usage recommendation**
>
> Use **Groq** for quick queries (free, response in <1s) and **Claude Code** when you need the AI to modify files or run commands on your machine.

### Difference between Claude Code and Anthropic API

- **Claude Code** ( `/ia claude` ) — Full agentic mode. Can read files, write code, run commands, navigate your project. Uses the Claude Code CLI.
- **Anthropic API** ( `/ia anthropic` ) — Chat only. Answers questions but has no access to your file system. Uses the API directly.

# 3. Installation

## 3.1 Automatic installation (recommended)

A single command that verifies requirements, downloads, installs, and configures:

```
bash installer.sh
```

The installer does everything automatically:

1. Verifies Node.js 20+, npm, git, Claude Code CLI
2. Downloads or updates the project
3. Installs npm dependencies
4. Launches the interactive configurator (6 steps)
5. Optionally creates an auto-start service

## 3.2 Manual installation

```
# 1. Clone repository
git clone <repo-url> ~/llm-remote
cd ~/llm-remote

# 2. Install dependencies
npm install

# 3. Configure
npm run setup

# 4. Start
npm start
```

## 3.3 Requirements

| Requirement | Version | Required | Notes |
|---|---|---|---|
| Node.js | 20+ | Yes | `brew install node` or nodejs.org |

| | | | |
|---|---|---|---|
| npm | any | Yes | Comes with Node.js |
| git | any | Yes | `xcode-select --install` (macOS) |
| Claude Code CLI | any | No | `npm i -g @anthropic-ai/claude-code` |

> **Without Claude Code CLI**
>
> If you don't install Claude Code CLI, the `/ia claude` provider won't work. The other providers (OpenAI, Gemini, Groq, Anthropic) will work fine without it.

# 4. Configuration

## 4.1 Create a Telegram bot

1. Open @**BotFather** in Telegram
2. Send `/newbot`
3. Choose a name (e.g.: *"My LLM Remote"*)
4. Choose a username (e.g.: *"my_llm_remote_bot"*)
5. Copy the **token** BotFather gives you

## 4.2 Get your Telegram User ID

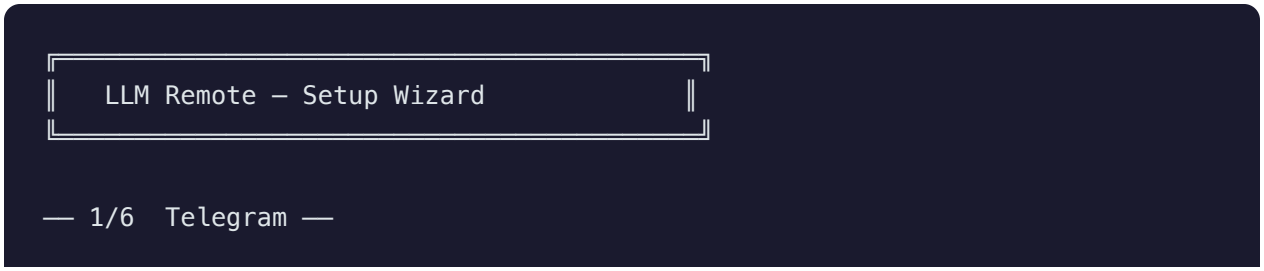
1. Open @**userinfobot** in Telegram
2. Send `/myid`
3. Copy the number (e.g.: *123456789*)

## 4.3 Get API keys (optional)

| Provider | URL | Cost |
|---|---|---|
| Gemini | https://aistudio.google.com/apikey | Free |
| Groq | https://console.groq.com/keys | Free |
| OpenAI | https://platform.openai.com/api-keys | Paid |
| Anthropic | https://console.anthropic.com/settings/keys | Paid |

## 4.4 Configuration wizard

The wizard ( `npm run setup` ) guides you through 6 steps:



```
  Bot token: <paste token>
  Authorized IDs: <your user ID>

── 2/6  Security ──
  PIN: <auto-generated or custom>
  Master password: <auto-generated>

── 3/6  Session & Limits ──
  Timeout: 15 min
  Max commands/min: 10

── 4/6  Claude Code CLI ──
  Binary: claude
  Directory: /Users/your-username

── 5/6  AI Providers ──
  OpenAI, Gemini, Anthropic, Groq (optional)

── 6/6  Logging ──
  Level: info
```

**Reconfigure at any time**

Run `npm run setup` again. Current values appear as defaults.

# 5. Usage from Telegram

## 5.1 First use

1. Start the bot: `npm start` (or the service does it automatically)
2. Open your bot in Telegram
3. Send `/start` to see available commands
4. Authenticate: `/auth <your-PIN>`
5. The PIN message is automatically deleted for security
6. Type any message — it goes directly to the active AI provider

## 5.2 Switch AI provider

```
# View available providers
/ia

# Switch to Groq (free, ultra-fast)
/ia groq

# Switch to Claude Code (agentic)
/ia claude

# Switch to Gemini (free)
/ia gemini
```

## 5.3 Working with projects

```
# View current directory
/project

# Switch to another project
/project ~/my-project

# Now Claude Code commands operate in ~/my-project
Refactor the main.js file
```

## 5.4 Sessions and security

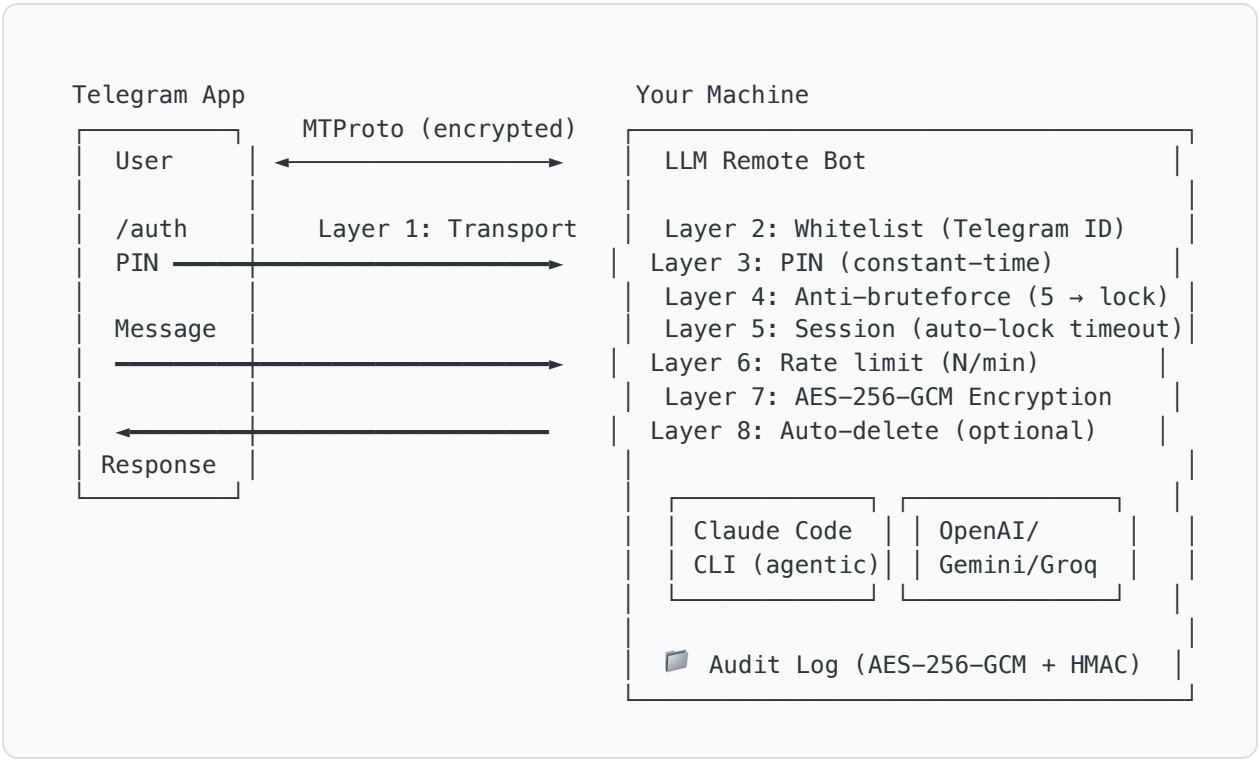
```
# View session status
/status

# Lock manually
/lock

# Re-authenticate
/auth <PIN>

# View history (decrypted)
/history
```

# 6. Command Reference

| Command | Description | Example |
|---|---|---|
| `/start` | Shows help and available providers | `/start` |
| `/auth <PIN>` | Authenticate. Message is auto-deleted. | `/auth <your-PIN>` |
| `/ask <prompt>` | Send explicit prompt to active provider | `/ask explain what this regex does` |
| `/ia [name]` | View or switch AI provider | `/ia groq` |
| `/project [path]` | View or change working directory | `/project ~/app` |
| `/status` | Session, provider and process info | `/status` |
| `/history` | Last 15 commands (decrypted from audit log) | `/history` |
| `/kill` | Kill running Claude Code process | `/kill` |
| `/lock` | Lock session immediately | `/lock` |
| `/help` | Show all commands | `/help` |
| *(free text)* | Sent directly to the active provider | *How do I optimize this SQL query?* |

# 7. Security Architecture

## 7.1 8-layer model




## 7.2 Encryption

| Component | Specification |
|---|---|
| Algorithm | AES-256-GCM (authenticated encryption) |
| Key derivation | PBKDF2 with 310,000 iterations + SHA-512 |
| IV (initialization vector) | 16 random bytes per message |
| Salt | 32 random bytes per message |
| Integrity | HMAC-SHA256 over the entire payload |
| Result | Each encryption is unique, even with the same plaintext |

## 7.3 Authentication

- **Whitelist** — Only configured Telegram IDs can interact. Unauthorized users are silently ignored.
- **PIN** — Constant-time comparison to prevent timing attacks.
- **Anti-bruteforce** — 5 failed attempts → 15-minute lockout.
- **Sessions** — Auto-lock after configurable inactivity (default: 15 min).
- **Auto-delete** — The PIN message is automatically deleted from Telegram.

> **Important: Per-installation security**
>
> Each installation generates its own master encryption password. Keys are not shared between users. If you lose your `.env`, there is no way to recover audit log data.

# 8. Technical Architecture

## 8.1 Technology stack

| Component | Technology | Rationale |
|---|---|---|
| Runtime | Node.js 20+ | Native async/await, native fetch, no native dependencies |
| Telegram bot | grammY | Modern framework, TypeScript-ready, middleware |
| Encryption | node:crypto | Native module, AES-256-GCM, PBKDF2, HMAC |
| AI APIs | native fetch | Zero dependencies, standard HTTP |
| Configuration | dotenv | Industry standard |
| Audit log | Encrypted NDJSON | Append-only, no database, portable |

## 8.2 Project structure

```
llm-remote/
├── src/
│   ├── index.js              # Entry point
│   ├── bot.js                # Telegram bot + handlers
│   ├── setup.js              # Interactive configurator
│   ├── auth/
│   │   ├── guard.js          # Whitelist + anti-bruteforce
│   │   └── session.js        # Sessions + timeout
│   ├── crypto/
│   │   └── cipher.js         # AES-256-GCM + HMAC + PBKDF2
│   ├── providers/
│   │   ├── base.js           # Base interface
│   │   ├── manager.js        # Multi-provider manager
│   │   ├── claude.js         # Claude Code CLI
│   │   ├── openai.js         # OpenAI GPT-4o
│   │   ├── gemini.js         # Google Gemini
│   │   ├── groq.js           # Groq Llama 3.3
│   │   └── anthropic.js      # Anthropic Sonnet
│   ├── claude/
│   │   └── formatter.js      # Chunking for Telegram
│   ├── security/
```

```
│   │   ├── ratelimit.js      # Rate limiting
│   │   └── audit.js          # Encrypted audit log
│   └── utils/
│       ├── config.js         # Configuration
│       ├── logger.js         # Logger
│       └── keygen.js         # Key generator
├── tests/
│   └── crypto.test.js        # 11 encryption tests
├── installer.sh              # Corporate installer
├── .env.example              # Config template
└── package.json
```

## 8.3 Dependencies

Only **2 production dependencies** (zero native):

| Package | Version | Purpose |
|---|---|---|
| grammy | ^1.31 | Telegram bot framework |
| dotenv | ^16.4 | Environment variables |

# 9. Troubleshooting

| Problem | Cause | Solution |
|---|---|---|
| Bot doesn't respond | Not running or wrong token | Check `npm start` and the token in `.env` |
| "Session expired" | Inactivity timeout | Send `/auth <PIN>` again |
| "Locked out" | 5 failed PIN attempts | Wait 15 minutes |
| "Rate limited" | Too many commands per minute | Wait a few seconds |
| "Provider not configured" | Missing provider API key | Run `npm run setup` and add the API key |
| Claude Code not working | CLI not installed | `npm i -g @anthropic-ai/claude-code` |
| Gemini error 429 | Daily limit reached (20 req) | Switch to Groq: `/ia groq` |
| Encryption error | .env corrupted or changed | Previous audit log will be unreadable. Delete `data/` and start over. |

## Logs

```
# View logs in real time
tail -f ~/llm-remote/data/llm-remote.log

# If using macOS service
tail -f ~/llm-remote/data/llm-remote.log

# If using Linux service
journalctl --user -u llm-remote -f
```

## Reconfigure

```
cd ~/llm-remote && npm run setup
```

## Uninstall

```
bash ~/llm-remote/installer.sh --uninstall
```

# 10. Environment Variables

| Variable | Req. | Default | Description |
|---|---|---|---|
| `TELEGRAM_BOT_TOKEN` | Yes | — | Telegram bot token |
| `AUTHORIZED_USERS` | Yes | — | Authorized Telegram IDs (comma-separated) |
| `AUTH_PIN` | Yes | — | Authentication PIN |
| `MASTER_PASSWORD` | Yes | — | Master encryption password (16+ chars) |
| `SESSION_TIMEOUT_MIN` | No | 15 | Session timeout (minutes) |
| `RATE_LIMIT_PER_MIN` | No | 10 | Max commands/minute |
| `AUTO_DELETE_SEC` | No | 0 | Auto-delete messages (0 = off) |
| `CLAUDE_BIN` | No | claude | Claude Code binary path |
| `DEFAULT_WORK_DIR` | No | $HOME | Default working directory |
| `MAX_CONCURRENT` | No | 2 | Concurrent Claude processes |
| `OPENAI_API_KEY` | No | — | OpenAI API key |
| `OPENAI_MODEL` | No | gpt-4o | OpenAI model |
| `GEMINI_API_KEY` | No | — | Google Gemini API key |
| `GEMINI_MODEL` | No | gemini-2.5-flash | Gemini model |
| `ANTHROPIC_API_KEY` | No | — | Anthropic API key |
| `ANTHROPIC_MODEL` | No | claude-sonnet-4 | Anthropic model |
| `GROQ_API_KEY` | No | — | Groq API key |
| `GROQ_MODEL` | No | llama-3.3-70b | Groq model |
| `LOG_LEVEL` | No | info | debug/info/warn/error |

LLM Remote v1.3 · Redegal · Digital Consulting Group · February 2026