LLM Remote

Encrypted Telegram ↔ AI Multi-Provider Bridge

REDEGAL - A SMART COMPANY

Version 2.1 · 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 can you do?

- **Remote development** — Run Claude Code on your machine from anywhere
- **Quick queries** — Ask GPT-4o, Gemini, or Llama from your phone
- **Voice notes** — Send audio and receive transcription + AI response
- **Visual analysis** — Send photos and screenshots for AI analysis
- **Files** — Send code, CSV, PDF and the AI processes them
- **Web search** — Search the internet and get an AI summary
- **Pipelines** — Chain operations: step1 → step2 → step3
- **Scheduled tasks** — Run prompts periodically
- **SSH remote** — Execute commands on remote servers from Telegram
- **Text-to-Speech** — Receive responses as voice notes
- **Groups** — Use the bot in Telegram groups
- **MCP** — Connect external tools via Model Context Protocol
- **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 + Vision	Pay-per-use
🔵 **Gemini 2.5 Flash**	`/ia gemini`	Chat API + Vision	Free 20 req/day
🟠 **Groq Llama 3.3**	`/ia groq`	Chat API (ultra-fast) + Whisper TTS	Free 30 req/min
🟣 **Anthropic Sonnet**	`/ia anthropic`	Chat API + Vision (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:

```
╔══════════════════════════════════════════╗
║   LLM Remote — Setup Wizard              ║
╚══════════════════════════════════════════╝

── 1/6  Telegram ──
```

```
  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 Conversational context

LLM Remote keeps a history of up to **20 messages** per user. The AI remembers what you discussed before.

```
# The AI remembers previous messages
Explain what a closure is in JavaScript
# ... response ...
Give me a practical example
# ... the AI knows you're talking about closures ...

# Clear context when switching topics
/clear
```

5.5 Sessions and security

```
# View session status
/status

# Lock manually
/lock

# Re-authenticate
/auth <PIN>

# View history (decrypted)
/history
```

6. Multimedia: Voice, Photos, Files

6.1 Voice notes

Send an audio or voice note to Telegram. The bot:

1. Transcribes the audio with **Groq Whisper** (free) or OpenAI Whisper
2. Displays the transcription
3. Sends the text to the active AI provider
4. Returns the response (and optionally as a voice note with TTS)

Perfect for mobile

Record an audio: "Explain how async/await works in JavaScript" and receive the response in text. Enable `/voz` to also receive it as a voice note.

6.2 Text-to-Speech (TTS)

Enable voice mode to receive responses as voice notes in addition to text:

```
# Toggle voice mode
/voz

# Now all responses include a voice note
Explain what Docker is in 2 sentences
# You receive text + audio
```

Supported TTS providers: **OpenAI TTS** (nova voice, opus format) and **Groq TTS** (PlayAI). The first available one is used.

6.3 Photo analysis

Send a photo or screenshot. The AI analyzes it with Vision:

- **OpenAI GPT-4o Vision** (priority)
- **Anthropic Claude Vision** (fallback)
- **Google Gemini Vision** (fallback)

```
# Send a photo with optional caption
[photo] What error is in this code?

# Or send without caption for general analysis
[photo]
```

6.4 File processing

Send code, text, data, or PDF files. The bot extracts the content and sends it to the AI.

Type	Extensions
Code	.js, .ts, .py, .go, .rs, .java, .c, .cpp, .rb, .php, .swift, .kt, .sh, .bash
Data	.csv, .json, .yaml, .yml, .xml, .toml
Text	.md, .txt, .log, .html, .css, .sql, .env, .conf, .ini, .cfg
Documents	.pdf (basic text extraction)

Limit: **5 MB** per file. CSV shows the first 50 rows as preview.

7. Advanced Tools

7.1 Web search

Search the internet and get an AI summary. No API key needed (uses DuckDuckGo).

```
# Search and summarize
/web Node.js 22 new features

# Search news
/web latest AI news 2026
```

7.2 Scheduled tasks

Run prompts automatically at regular intervals.

```
# Create a task every 24 hours
/schedule 24h Summarize the repo status

# Create a task every hour
/schedule 1h Check for errors in the logs

# Supported intervals: 5m, 30m, 1h, 6h, 24h, 7d

# List tasks
/schedules

# Delete task
/unschedule 3
```

7.3 Pipelines

Chain multiple operations where the output of one step feeds into the next:

```
# Search + summarize
/pipe search React 2026 trends → summarize in 3 key points

# Analyze + suggest + draft
/pipe read the project README → suggest improvements → draft a PR
```

```
# Data + statistics + format
/pipe analyze this CSV → generate statistics → format as markdown table
```

Valid separators: `→`, `|`, `>`

7.4 MCP servers

Connect external tools via **Model Context Protocol**:

```
# Add MCP server
/mcp add github npx -y @modelcontextprotocol/server-github

# View available tools
/mcp tools

# Execute tool
/mcp call github/list_repos {"owner": "redegal"}

# View connected servers
/mcp

# Remove server
/mcp remove github
```

8. SSH Remote

Execute commands on remote servers directly from Telegram. Uses native SSH (no additional dependencies).

8.1 Configure servers

```
# Add server
/ssh add prod root@37.27.92.122
/ssh add staging deploy@staging.example.com 2222

# Full format
/ssh add <name> <user@host> [port] [ssh-key-path]

# List servers
/ssh list

# Remove server
/ssh remove prod
```

8.2 Execute commands

```
# Disk space
/ssh prod df -h

# Docker container status
/ssh prod docker ps

# Last log lines
/ssh prod tail -20 /var/log/syslog

# System status
/ssh prod uptime && free -h
```

8.3 SSH security

Blocked commands

The following commands are blocked for safety:

> - `rm -rf /` (root deletion)
> - `mkfs` (format disk)
> - `dd if=` (direct disk write)
> - `shutdown`, `reboot`, `init 0`

Other security measures:

- **30-second** timeout per command
- Output truncated to 4000 characters
- `BatchMode=yes` (no interactive prompts)
- All commands are logged in the audit log
- Persistent configuration in `data/ssh-servers.json`

9. Telegram Groups

LLM Remote works in Telegram groups and supergroups. In groups, the bot only responds to:

- **Commands** — `/status`, `/ia groq`, etc.
- **Mentions** — `@my_bot Explain this`
- **Replies to the bot** — Reply to a bot message
- **Media** — Photos, audio, and files sent in the group

Regular group messages are ignored to avoid cluttering the conversation.

Group setup

1. Add the bot to the group
2. All authorized users (whitelist) can use it
3. Unauthorized users are silently ignored

10. Command Reference

Command	Description	Example
`/start`	Shows help and available providers	`/start`
`/auth <PIN>`	Authenticate. Message is auto-deleted.	`/auth 1234`
`/ask <prompt>`	Send explicit prompt to active provider	`/ask explain this regex`
`/ia [name]`	View or switch AI provider	`/ia groq`
`/clear`	Clear conversational context (20 msgs)	`/clear`
`/project [path]`	View or change working directory	`/project ~/app`
`/status`	Session, provider, TTS, SSH info	`/status`
`/history`	Last 15 commands from audit log	`/history`
`/kill`	Kill running process	`/kill`
`/lock`	Lock session immediately	`/lock`
`/voz`	**New** Toggle voice responses (TTS)	`/voz`
`/web <query>`	Web search + AI summary	`/web React 2026 news`
`/schedule`	Create scheduled task	`/schedule 24h Daily summary`
`/schedules`	List scheduled tasks	`/schedules`
`/unschedule <id>`	Delete scheduled task	`/unschedule 3`
`/pipe`	Execute step pipeline	`/pipe search X → summarize`
`/mcp`	Manage MCP servers	`/mcp add github npx ...`
`/ssh`	**New** SSH remote: add, list, remove, execute	`/ssh prod df -h`

`/help`	Show all commands	`/help`
(free text)	Sent directly to the active provider	*How do I optimize this SQL query?*
(audio)	Transcription + AI processing	*[voice note]*
(photo)	Visual analysis with Vision	*[screenshot]*
(file)	Content extraction + AI analysis	*[file.csv]*

11. Security Architecture

11.1 8-layer model

```
  Telegram App                          Your Machine
 ┌──────────┐      MTProto (encrypted)   ┌───────────────────────────────────────┐
 │  User    │ ◄───────────────────────► │ LLM Remote Bot v2.1                   │
 │          │                            │                                       │
 │  /auth   │      Layer 1: Transport    │  Layer 2: Whitelist (Telegram ID)     │
 │  PIN     │──────────────────────►    │ Layer 3: PIN (constant-time)          │
 │          │                            │  Layer 4: Anti-bruteforce (5 → lock)  │
 │  Message │                            │  Layer 5: Session (auto-lock timeout) │
 │          │──────────────────────►    │ Layer 6: Rate limit (N/min)           │
 │          │                            │  Layer 7: AES-256-GCM Encryption      │
 │ ◄────────│───────────────────        │ Layer 8: Auto-delete (optional)       │
 │ Response │                            │                                       │
 └──────────┘                            │  ┌─────────────┐ ┌──────────────┐    │
                                         │  │ Claude Code │ │ OpenAI/Gemini│    │
                                         │  │ (agentic)   │ │ Groq/Anthro  │    │
                                         │  └─────────────┘ └──────────────┘    │
                                         │                                       │
                                         │  Audit Log (AES-256-GCM + HMAC)       │
                                         └───────────────────────────────────────┘
```

11.2 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

11.3 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.

12. Technical Architecture

12.1 Technology stack

Component	Technology	Rationale
Runtime	Node.js 20+	Native async/await, native fetch, no native deps
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
Transcription	Groq Whisper / OpenAI Whisper	Groq free, OpenAI fallback
Vision	GPT-4o / Claude / Gemini	Triple fallback for max availability
TTS	OpenAI TTS / Groq TTS	Dual fallback
SSH	native ssh (child_process)	Zero dependencies, uses system keys
Web search	DuckDuckGo scraping	No API key, free
Configuration	dotenv	Industry standard
Audit log	Encrypted NDJSON	Append-only, no database, portable

12.2 Project structure

```
llm-remote/
├── src/
│   ├── index.js          # Entry point
│   ├── bot.js            # Telegram bot + handlers
│   ├── setup.js          # Interactive configurator
│   ├── auth/             # Whitelist, sessions, groups
│   ├── crypto/           # AES-256-GCM + HMAC + PBKDF2
│   ├── providers/           # Claude, OpenAI, Gemini, Groq, Anthropic
│   ├── context/             # Conversational memory (20 msgs)
│   ├── media/               # Voice, Vision, Files, TTS
│   ├── search/              # Web search (DuckDuckGo)
│   ├── scheduler/           # Scheduled tasks
│   ├── pipeline/            # Pipeline engine
│   ├── mcp/                 # MCP client (JSON-RPC stdio)
│   ├── remote/              # SSH remote + safety
│   ├── claude/              # Telegram message chunking
│   ├── security/            # Rate limiting + encrypted audit
│   └── utils/               # Config, logger, keygen
├── tests/                   # 53 tests (7 suites)
├── docs/                    # Manuals ES/EN (HTML + PDF)
├── installer.sh             # Corporate installer
└── package.json
```

12.3 Dependencies

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

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

13. 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`
TTS not working	No OpenAI or Groq API key	Configure at least one OpenAI or Groq API key
Voice not transcribing	No Groq or OpenAI API key	Groq is free: get a key at console.groq.com
SSH timeout	Server unreachable or wrong SSH key	Verify connectivity and SSH key configuration
SSH "command blocked"	Dangerous command detected	Security protection, cannot be disabled
Bot not responding in group	Only responds to commands/mentions	Mention @bot or reply to a bot message
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 Linux service
journalctl --user -u llm-remote -f
```

Tests

```
# Run all 53 tests
npm test
```

Reconfigure

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

Uninstall

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

14. 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 (chat + vision + TTS)
`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 (chat + whisper + TTS)
`GROQ_MODEL`	No	llama-3.3-70b	Groq model
`LOG_LEVEL`	No	info	debug/info/warn/error