

ENIGMA 3

Your messages. Your server. Nobody else's.

Over 60 million messages were read by someone else. EncroChat, Sky ECC and ANOM proved why — not the encryption failed, the **operator** did. Enigma 3 removes him from the picture: your messages, your server, nobody else's. End-to-end encrypted on every layer — no corporation, no cloud, no telemetry.

🔒 End-to-end encrypted

🏠 100 % self-hosted

⏱️ Self-destructing messages

📞 Encrypted calls

🖥️ Windows · Linux · Android · Web

WHY ENIGMA 3

— Trust nobody. Least of all a provider.

Three of the world's "unhackable" services failed at a single point — not the encryption, the operator. The numbers come from official Europol, Eurojust and prosecution records.

ENCROCHAT · 2020

60M

messages read · €700M seized · **6,500+ arrests**

60,000 devices sold as "absolutely secure". Police didn't crack the crypto: through the **vendor's own update program** every device got a spyware build. Users never knew their conversations had been running in the open for years — long before officers started reading them.

SKY ECC · 2021

€700M

seized · dozens of arrests on three continents

Servers seized in France and Canada, founder arrested. The decisive difference to EncroChat: **the provider held the keys itself** — and handed them to the authorities. A "secure" system whose manufacturer turns helpful, key book and all.

ANOM · 2021

300+

convictions · 40 tons of drugs seized

No bug, no hack — **the product was the investigation**. In Operation Trojan Shield, run covertly by the FBI, every message ran through servers owned by investigators for two years. Buying ANOM isn't buying protection — it's renting an evidence machine.

The common failure: never the encryption — **always the operator.** Whoever holds the server holds the keys, without breaking a single algorithm.

The everyday comparison

	WhatsApp	Telegram	Signal	ENIGMA 3
Server belongs to	Meta	the provider	Foundation (USA)	**you**
Read access	provider (optional backup)	provider (in "cloud chats")	nobody — except your own backups	**nobody**
Metadata	at Meta	at the provider	at Signal, minimized	**on your hardware**
Identity	phone number	phone number or username	phone number required	**username, no phone forced**
Quitting	provider deletes your data	provider deletes your data	deletes after 30 days of inactivity	**you are the operator**

The principle: Enigma 3 has no vendor instance to take over, no update program to abuse, no keys held by someone else. The seizure that was everything at EncroChat is, here, an empty hard drive.

WHAT TRUST COSTS

— Privacy has a price — honest numbers

	WHATSAPP / TELEGRAM	TEAMS / SLACK	ENIGMA 3
What you pay	0 € — with your data, your reach and your habits	A subscription per person, per month	**Once: your own server. No subscription, no per-seat billing.**
Who owns the server	the provider	the provider	**you**
Who reads the content	the provider — whenever they want	the provider — whenever they want	**nobody. Only your devices.**
Metadata	at the provider, with real names	at the provider, with real names	**at you. The server only sees who wrote to whom, when.**
If access is blocked	gone — the provider decides	gone — the provider decides	**you decide. Always.**

🤝 Your operator — without the trust gap

Setup, updates and backups are handled by your operator — you just chat. And here is the guarantee that makes it safe: **even the operator cannot read a single message, key or password.** It is technically impossible — the keys live on your devices, the server only ever sees ciphertext. What the operator sees is the connection, never the content.

📜 Data protection you can show the authorities

Server sovereignty with you, data processing agreements self-managed, no data transfer to foreign countries, no corporate terms you can't control. **GDPR is not a promise here — it is the architecture.** And you can leave at any time: server wiped, keys destroyed, nothing remains. No lock-in.

HOW IT FEELS IN EVERYDAY LIFE

Three groups. Three good reasons.

👨‍👩‍👧‍👦 The family group

Photos of the grandchildren, holidays, the difficult conversations — they stay in the family's own archive. Not read, not used for ads, not “just for delivery". The server belongs to the house, not to a corporation.

💼 The company

Projects, offers, calculations — on a server one person can see through, not in a cloud whose backups end up with other companies. Confidential here means: **at your place** — not at a privacy board that approves in case of doubt.

⚖️ Confidential work

Practice, newsroom, club, business — whoever guards secrets doesn't outsource them. With Enigma 3, only your server decides who may read anything. And that server stands at your place, not at Meta's, not at Microsoft's, not in a data centre whose customer list you don't know.

What a seizure gets you

🔒 The day the police come

Imagine your server is seized — tomorrow, without warning. What ends up on the table: **terabytes of ciphertext. Not one readable sentence, no channel name, no profile name, no reaction.** The keys live on your devices, not on the server — and because the server never had them, it cannot hand them over. What remains is the bare connection: who spoke to whom, when. What was said: never.

And the devices don't stay silent: **a new device, a password reset — every incident alerts all other devices of the account.** Whoever wasn't you is revoked with the stroke “That wasn't me”. An attacker trying to take over your account becomes visible the same day. EncroChat's users had the opposite experience — thousands of officers read everything without anyone noticing. The difference isn't luck. It's whose server.

CRYPTOGRAPHY, NOT TRUST

— “Unbreakable” — why the word is not marketing here

There are two ways to sell security: one you can verify — and one you have to believe.

🧮 The math

AES-256 has **2^{256} possible keys** — more variants than the universe has particles. Brute force is not expensive; it is physically impossible: every computer on Earth for the lifetime of the sun — no hit. That is why states protect their secrets with it: not out of courage, out of math.

🔬 The scrutiny

X3DH, Double Ratchet, AES-256-GCM — not an invention you must trust, but decades of public work: thousands of cryptographers, standard documents, deployed in Signal, iMessage and banking. Kerckhoffs’ rule: security must work even when everyone knows the method. With Enigma 3 the design is openly documented — safety hangs on the key, not on the method.

⚠️ The break point — and why it is missing here

AES has never been cracked, X3DH broken, the Double Ratchet overcome. **Every case of read chats — EncroChat, Sky ECC, ANOM — failed not at the math but at the edges:** an operator handing out keys, an update program turned spyware, a product that was itself the backdoor. Enigma 3 removes those edges: server at your place, keys on your devices, an alert on every new device, revocation on every reset. What remains is the math alone.

What “unbreakable” does not mean: the metadata (who spoke to whom, when), your own device, and the legal situation if you are compelled to hand over your keys. Everything else — content, names, reactions, attachments — stays ciphertext.

COMPARED

— Against the field — Signal, Matrix, Session, SimpleX

All four are good messengers. The difference is not the algorithm — they share it — but who owns the servers, what remains after deletion, and who can attack in advance.

	Signal	Matrix	Session	SimpleX	**Enigma 3**
Server belongs to	Foundation (USA)	operators, federated	Oxen consortium	server operator	**you**
Access	phone number required	e-mail, server free	seed phrase	no identity	**username + invite code**
Real deletion	server retention	soft-mask, operator must comply	after delivery	after delivery	**crypto-shredding, backups too**
Timer extendable?	server sets deadline	operator sets deadline	—	—	**no — it lives in the ciphertext**
Silent readers?	—	server can add members quietly	—	—	**sending pauses until confirmed**

🎯 No mass target — and no map

Anyone attacking Signal, WhatsApp or Telegram invents nothing — it is already built: one tool for one of a hundred million **identical** devices, and everyone is hit. That is how EncroChat and Sky ECC fell. **No such toolbox exists for Enigma 3.** The code is not public, it runs nowhere a million times over — nobody whom authorities could borrow their toolkit from has ever turned this code under real users. There is no map where an attacker could start. A prepared attack runs into empty air: it must be made singly, from zero, against exactly your server — no head start, no hit list.

Honest, not advertised: your operator sees the connection — who, when, where. Never what you send. Everything else is a promise you can verify.

SECURITY

Encrypted to the roof — as the default

You don't configure anything. All of the following applies from the very first message:

🔐 End-to-end encryption

Direct messages: X3DH + Double Ratchet (like Signal) with forward secrecy, self-healing and replay protection. **Channels and groups:** sender keys, rotated on every departure. **Attachments:** AES-256-GCM, filename and type inside the ciphertext. The server stores and relays **ciphertext only** — it cannot read anything.

📞 Encrypted calls — on your server

Voice and video with screen sharing, a media key per call, packed to the participants' device keys. New key at every participant change. **The LiveKit server runs at your place** — what Signal and WhatsApp do in their data centers happens here on your hardware. Can run without the LiveKit container if you prefer.

⏱️ Self-destructing messages

Lifetime per message: off · 1 h · 7 h · 24 h · 3 · 7 · 30 days (default 7). The timer lives **inside the encrypted content** — the server cannot extend it. The client deletes at the earlier of its own timer or the server's time and shows deviations.

🔥 Irreversible deletion

Crypto-shredding: every message has its own key. Deleting or expiry destroys the key — copies on the server and in backups become unreadable. Deliberately **no built-in backups** or snapshots that would defeat deletion.

📱 Devices & safety numbers

A separate key per device, never leaving it. Review and revoke devices, alerts on new devices, optional auto-removal of inactive devices. Safety numbers for comparison rule out unnoticed eavesdropping. **A new device cannot read older messages** — those keys were never created for it.

🔑 Local vault

Biometrics or a PIN seal the keys on the device (Argon2id, AES-GCM). The PIN is eight digits and mandatory on desktop. Android Keystore/StrongBox, iOS Secure Enclave, Windows DPAPI. Without the PIN, a seized device yields only ciphertext.

🔐 Two-factor sign-in (TOTP)

Beyond the password, a six-digit one-time code from an authenticator app — set up under Settings → Security. Firm **backup codes** help when the device is out of reach. Sign-in happens with **username and password** alone — the system stores and sends no e-mail addresses whatsoever; access comes from the operator's invitation code.

CRYPTOGRAPHY

How the encryption works — in detail

Vetted standard primitives, no proprietary crypto: NIST P-256, HKDF/HMAC-SHA-256, AES-256-GCM. Private keys never leave the device.

🤝 Key agreement: X3DH

The sender fetches the recipient device's **prekey bundle** and verifies the signed prekey against the **pinned identity signing key** (trust on first use). The result is a shared secret known only to these two devices. Prekey retrieval is rate-limited against draining one-time prekeys.

🔁 Double Ratchet (DMs)

Every message gets its **own key**. The symmetric ratchet gives forward secrecy; the Diffie-Hellman ratchet gives **self-healing** after a compromise. Headers are encrypted. Seen base keys are rejected (replay protection). Each message is encrypted for **every recipient device** (up to 64) and your own other devices. A device that joins later **cannot read older messages** — those keys were never created for it.

👥 Sender Keys (groups & channels)

Each device maintains a chain per room: `mk = HMAC(ck, 0x01), ck' = HMAC(ck, 0x02)`. Every message is signed with the chain's signing key; the signature covers room and sender. Chains are distributed **through the pairwise Double-Ratchet sessions** to confirmed members only. Rotation: on any departure, device removal, device change, after 7 days, or before counter overflow. New members get the chain from the current state — **older messages stay unreadable**.

📞 Calls & attachments

Calls: LiveKit Insertable Streams. The devices agree on the call key themselves and pack it to each participant's device keys; new key at every change. The server never generates, stores or outputs one. **Attachments:** AES-256-GCM via the ratchet (DMs) or sender key (channels), bound to room and sender. EXIF, GPS and XMP metadata are stripped on upload.

Keys per device

Identity key (ECDH P-256)	Device identity, basis of the safety number — permanent per device
Signing key (ECDSA P-256)	Signs the signed prekey; pinned on first contact (TOFU)
Signed prekey & one-time prekeys	For the X3DH exchange — renewed regularly, replenished after use, retrieval rate-limited

ENFORCEMENT & TRUST

What the server refuses — and what the client watches

🚫 The server rejects plaintext

`CreateMessage`, `UpdateMessage`, `CreateUpload` and image descriptions return `e2ee_required` without ciphertext — in DMs, groups and channels. This duty **cannot be switched off in production**, neither by setting nor environment variable. Old plaintext channel messages are destroyed at startup: key destroyed, storage securely overwritten.

🚪 No silent eavesdroppers

The client remembers the first seen member list. Members added later stay **pending** until you confirm them in the room or verify their safety number. The server can claim new members — but cannot hand out keys unnoticed. In DMs, sending pauses until you confirm a changed key.

⚠️ Device warnings (TOFU)

Device keys are pinned on first contact. **A new device** only shows a “New device” note with a confirm button — **no alarm**. A real key change stays the red warning. Both pause sending until you confirm. New sign-ins alert **your other devices** with name and time — “Was that me?” — with one-click revocation.

🧹 Controlled deletion

System messages (joins, calls, room changes) are deletable immediately or automatically; the server securely overwrites stored events. After deletion only a content-free deletion proof remains. Only entries that define current state survive: room, name, settings and the last join per person.

🔒 Hard defaults

- ✓ **Invitation-only** registration
- ✓ **12-character** minimum password
- ✓ Login throttling with growing lockout, hard password lockout after repeated failures
- ✓ **Sign-in with username and password only** — optionally two-factor with one-time codes (TOTP) and backup codes. No e-mail addresses are stored or sent.
- ✓ **Absolute session lifetime** (default 12 h) — not just idle
- ✓ HTTPS with HSTS, strict CSP, no embedding
- ✓ Auto-removal of devices inactive 30–365 days (optional)

🧨 Removed, not disabled

- ✓ **No bots, no SSO, no webhooks** — no API-key door, no login through third parties, no automatic post router. These entry points would break end-to-end encryption, so the architecture was built without them
- ✓ **No link previews** — the server would see the URL in plaintext
- ✓ **No link previews** — the server would see the URL in plaintext
- ✓ **No telemetry, no crash reporting**
- ✓ **Metadata noise removed** — EXIF, GPS and XMP are stripped on image upload: a photo of the kitchen table keeps no location, no timestamp, no camera model. A picture you send is exactly that: a picture. Not a form.

DEVICE SECURITY

Keys under lock — the local vault

🔑 How the vault works

The PIN or passphrase derives a key-encryption key with **Argon2id** (256 MiB on desktop, existing vaults upgrade automatically; otherwise PBKDF2-SHA256 ≥ 600k iterations). It wraps a random 256-bit **data key (DEK)**, sealed together with a device-specific secret. Every vault entry is sealed with **AES-GCM** — unique nonces, the record ID as additional authenticated data. Platform binding adds a second layer: Android Keystore/StrongBox, iOS Secure Enclave, Windows DPAPI via safeStorage. On lock, inactivity timeout, background and app exit the DEK is discarded from memory. On mobile, every return from the background requires Face ID/Touch ID/fingerprint, PIN as fallback.

⛔ Attacks and answers

Wrong PIN: exponential backoff; optional vault wipe after N attempts (selectable: 3, 5, 10, 25 or 50; the “strong” level forces an 8-digit PIN and wipes after 5). A live estimate shows how fast an attacker with a copied database could test your PIN. **Forgotten PIN:** local wipe and re-registration as a new device — server messages stay. **Stolen database:** without the PIN only ciphertext; hardware binding makes offline brute force expensive. **Weak short PINs** stay offline-bruteforceable — a long passphrase is recommended.

📱 Device management

Settings show all devices with message keys; each is individually revocable. A revoked device is logged out and gets no new keys. Inactive devices can be auto-removed after 30, 60, 90, 180 days or a year — revoked completely along with keys, sessions and tokens.

✅ Safety numbers

Derived from both accounts' identity keys. Compare in person or via QR code — afterwards the person counts as **verified**. In groups and channels this works per member through the lock in the header, which also shows every member, their devices and the sender-key status. A key change on a verified device pauses sending until you confirm — in DMs, groups and channels alike.

THE SECURITY MODEL

Who sees what — and who cannot do what

Server operator / attacker with server access	Sees **metadata**: who with whom, when, which room, message sizes, IPs, device lists. **Cannot:** decrypt content, attachments or calls; extend timers; slip in members or devices unnoticed (if you check safety numbers).
Network observer	Sees a TLS connection. **Cannot** read content.
Stolen / seized device	With active PIN: keys and caches sealed (Argon2id/PBKDF2 KEK, AES-GCM DEK). Without PIN: ciphertext only; hardware binding makes brute force expensive. **Cannot** reconstruct older ratcheted messages of other devices.
Push provider	Sees time and target device. By default **no** plaintext titles or sender names — never content or sender names from the payload.
Investigator with server seizure	Memberships, timestamps, sizes, device lists, push tokens. **Without users' device keys: not a single readable message.**

What stays unencrypted — and why

- ✓ **Metadata** — communication partners, times, room membership, sizes. Inherent to any server architecture; state it openly.
- ✓ **Room names, profiles, reactions** — not covered by E2EE; protect server, `.env` and backups accordingly.
- ✓ **System messages** — generated by the server core (e.g. "… joined"); deletable, but not E2EE payloads.

Backup discipline: back up the VM **encrypted** and keep backups **no longer than the longest message lifetime you use** (default 7 days, max 30). Older backups would defeat deletion. Store `.env` separately — it contains the master key.

METADATA PRIVACY · STAGE 1

Encryption hides content — stage 1 takes the metadata too

The adversary here is the server operator or anyone who obtains a copy of the server data: event log, buckets, snapshots, backups, logs. Stage 1 removes the metadata the server does not need to operate and makes the rest less precise. Experimental, default-on, no configuration.

🏷️ Encrypted channel names

The client encrypts **channel name and description** with the room's sender keys; the server keeps only ciphertext and a neutral placeholder (🔒 `#4f2a9c01d7`). The plaintext name never reaches the server — renaming a migrated channel in plaintext is rejected (fail closed). Members who can read messages read the name; non-members and devices without the key see the placeholder. Old plaintext name records are deleted at migration.

📏 Size padding

Every encrypted body and edit is padded to a size bucket before encryption (Padmé rule, ≥256 bytes, at most ~12 % overhead, invisible marker characters). **The server cannot tell a short reply from a long sentence.** Attachments already used buckets.

⌨️ Typing opt-out

Settings → Security & privacy: “Send typing indicators” off means this device sends none — while you still see everyone else's. The choice is per device and server, never stored centrally.

🪪 Encrypted profile names

A display name and avatar are stored end-to-end encrypted; the server sees only ciphertext and a stable placeholder. Anyone who shares a readable room receives the grant key automatically — including devices that join later. Login names stay the visible constant.

😊 Encrypted reactions

New reactions travel the same path as the message itself: the server stores only an id and ciphertext and cannot tell how many reactions a message carries or which emoji they are. Reactions from the plaintext era stay readable and removable.

🧾 Minimal server logs

By default, process logs write **no user IDs, login or display names, room names, e-mails or IPs** — identity fields show as `[redacted]`. Sign-in audit records keep no hash of IP or user agent unless the operator enables it — and then only with a key that rotates on a window (default 24 h), so hashes cannot be chained across time. The session list still shows the coarse device class (“Firefox on Windows”).

CONTROL

— Your account, your server — with the hard buttons

🔑 Passkey sign-in (WebAuthn)

Sign in without a password: the server stores only the **public credential**, the private key stays on your device. On HTTPS servers passkeys are **on by default** — password sign-in always stays available and cannot be disabled. A **password reset warns every device of the account** — you see immediately whether someone used your account in the meantime. Each ceremony uses a five-minute nonce that verifies **at most once**. Adding or removing a passkey alerts your other sessions; removing revokes established credentials like a password change. The relying party is derived from the public HTTPS URL — credentials can never be scoped broader than your server.

📡 Active sessions: who is online

Settings → Security lists **every device signed into your account**: description, sign-in method, sign-in time, last use, an **online marker** and a “this device” marker. Online means a live realtime connection, tracked per session with 60-second heartbeats — not a guess. Every session except the current one can be **signed out with one click**. A stolen laptop or forgotten browser can be logged out precisely — without touching the account.

📱 Devices & keys under your hand

Settings → “Devices with message keys” shows every device of your account; each is **individually revocable**. A revoked device is logged out, loses sessions and tokens and gets **no new sender keys** — future messages skip it. New logins alert your other devices with name and time and the option “**That wasn’t me**” for one-click revocation. Operators can auto-remove devices inactive 30–365 days; abandoned devices are revoked completely, keys included.

Buttons for the hard cases

🗑️ Delete everything — the admin button

Administrators find a danger button in **Admin → Security → self-deleting messages**: “Delete all messages”. One action wipes **every message in every room, thread and DM** — with reactions, edits, attachments and link previews. The server reuses the expiry machinery: each message key shredded, body events erased, attachment bytes removed, projections updated, up to eight rooms in parallel as a background job with progress. **Accounts, rooms, memberships, devices, keys and settings stay.** Final, and in the audit log without message content.

🧹 Delete all *your* messages

Users have the same power over their own data: Settings → Self-Destructing Messages → “Delete all my messages”. The server retracts every post you wrote — including rooms you left, thread replies and their echoes — newest-first. Membership stays; content, reactions, pins, search entries, notifications, attachments and previews go. Crypto-shredded, not hidden.

⚙️ Admin → Security: inside the page

A dedicated **Security** page with three sections. **Access:** blocked usernames refused at registration; every OAuth client settable to default, trusted or blocked. **Retention:** “Delete all messages” danger button (background job with progress); system events — joins, calls, room changes — cleared immediately or following the message lifetime, securely overwritten. **Sessions:** absolute session timeout (e.g. 12 h, not just idle); **inactive devices auto-removed after 30, 60, 90, 180 days or 1 year** — revoked completely, keys, sessions and tokens, the last device of an account included; the panel shows how many devices are pending removal. Plus invitation links with expiry, password policy, login throttling, branding, audit log without content, IP-watch diagnostics.

CAPABILITIES

A messenger that does everything — encrypted

💬 Full-featured chat

Rooms, threads, groups and channels, mentions, reactions, Markdown, pinned messages, files — all end-to-end encrypted. Read receipts for direct messages, optional in both directions.

🖼️ Files & media

Images, documents, videos — encrypted, metadata stripped. Filename and type inside the ciphertext.

📞 Conferences

Voice and video with screen sharing, encrypted per call, keys only for confirmed members.

🔔 Notifications

Push carries only sender and location — never your text. UnifiedPush/FCM/APNs/Web Push supported.

🛡️ Verification

Safety numbers per person, device overview, alerts on new devices, optional auto-removal.

🧹 Controlled deletion

System messages immediately or automatically deletable — the server overwrites securely.

At home everywhere

🪟 **Windows 10/11 (x64)**	Installer; keeps all data and keys across updates.
🐧 **Linux x86_64 / arm64**	AppImage — make executable, double-click.
🤖 **Android 7+**	Signed release APK (`CN=Enigma 3`), biometric lock, Keystore vault.
📱 **iPhone / iPad**	Use the web app in Safari.
🌐 **Browser**	Web app served directly from your own server.

Updates: only from the official repository, checked against SHA-256 sums and a valid **Ed25519 signature** — without checksum and signature the app offers no update. Desktop and Android optionally support certificate pinning. **Integrity:** `sha256sum -c SHA256SUMS.txt`.

GETTING STARTED

Three steps to your own chat

1

Run the server

A Linux machine or VM. A single **setup.sh** walks you through the setup: domains, Cloudflare and Docker — no Docker knowledge needed, it asks for everything and configures it. You need your own domain. That's all the routine it takes.

2

Install the app

Desktop app, APK or web app — open “Connect server” and enter your address. Registration works by invitation link only, so the operator decides who joins.

3

Start securely

First direct message, compare the safety number, test the first call. If the call drops, the router ports are missing. Everything else is default.

Worth knowing: Enigma 3 is self-hosted software for people who want to take back control — not a free hosted service. You need your own Linux machine and a domain; setup and operation are handled by your operator. **Why does this exist?** Because we use it ourselves — no investors, no data-hungry business model behind it. **There is always only the latest version** — no outdated packages in circulation. Technical details, the security model and installation are fully documented in the open.

Switch. Keep talking.

Messages that stay with you — on hardware you own.