Down the Rabbit-Hole

A Forensic Analysis of the Matrix Protocol and Synapse Homeserver

Yikai Wang[1,2] · Xuepei Zhang[1] · Shufan Wu[1] · Yan Cheng[1]

[1] East China University of Political Science and Law | [2] Shanghai Jueyin Digital Forensic Institute

DFRWS

Presenter: Yikai wang | redundan3y@protonmail.com


...TUMBLING DOWN THE RABBIT HOLE?

What is Matrix???

Just think of it as an end-to-end messaging app / protocol for now.

OVERVIEW

Presentation Outline

01 **The War Story**

Game Leaks, Hackers and Forums

02 **Matrix Protocol Architecture / Background**

Federation, encryption (Olm & Megolm), key management

03 **Research Questions**

RQ1 & RQ2 — artifacts and timeline reconstruction

04 **Methodology**

Test environment, scenarios, and acquisition approach

05 **Server Log Artifacts**

Auth, messages, media, VoIP, crypto key events

06 **Database Artifacts**

175 tables — user IDs, relationships, content

07 **SynExtract Tool**

Architecture, capabilities, visualizations

08 **Discussion & Future Work**

Limitations and next steps

War story

War Story: Game Leaks, Hackers and Forums

Shanghai Jueyin Digital Forensic Institute

Shout out to them!

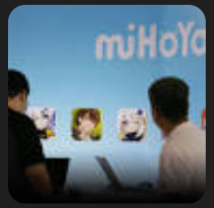
War story

War Story: Game Leaks, Hackers and Forums

Sixth Tone
https://www.sixthtone.com › news › shanghai-opens-first...

Shanghai Opens First Criminal Case Against Video Game ...

Mar 2, 2026 — Last April, the Supreme People's **Court** issued guidelines clarifying how **criminal** law applies to IP violations, including copyright infringement. Read more



In recent years, an increasing number of developers have turned to legal action to curb leaks. In 2024, a court in Chengdu, capital of China's southwestern Sichuan province, sentenced a man to three years in prison and fined him 300,000 yuan ($43,600) for leaking future character skins from gaming giant Tencent's "Honor of Kings."

https://www.sixthtone.com/news/1018250

**** Not this case exactly but similar ****

War story

War Story: Game Leaks, Hackers and Forums

公安机关查明，刘某某利用侵犯企业著作权的形式，不到半年时间账号粉丝数量飙升至70余万。经法院审理查明认定，4个月间，刘某某在网络平台发布《王者荣耀》“爆料视频”33个，获取点赞178万余次，获取平台结算的广告收益数十万元。

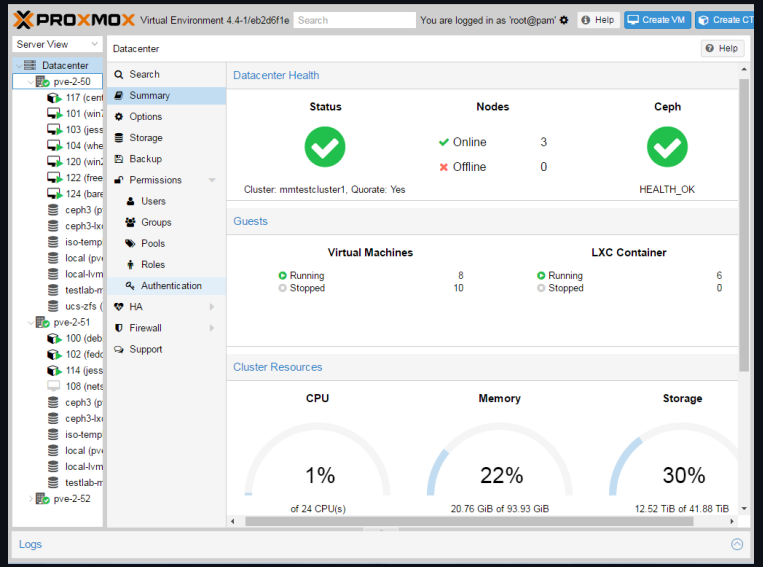
Police investigators determined that Liu used copyright-infringing content to grow his account to over 700,000 followers in less than six months.
Court proceedings further established that over a four-month period, Liu published 33 "leak videos" about *Honor of Kings* on online platforms, accumulating more than 1.78 million likes and earning hundreds of thousands of yuan in advertising revenue settled by the platforms.

https://mp.weixin.qq.com/s/-4rBqFafNygl6Uz2L8MK1A

**** Not this case exactly but similar ****

War story

War Story: Game Leaks, Hackers and Forums

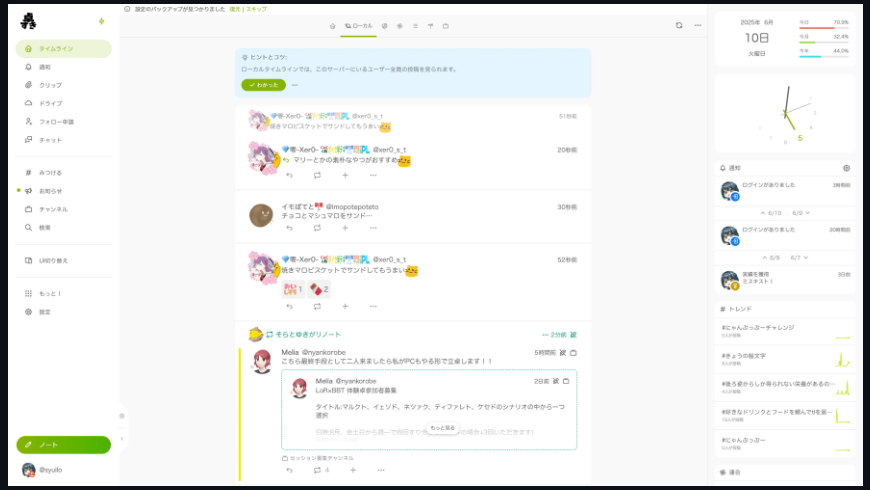


PROXMOX

www.proxmox.com

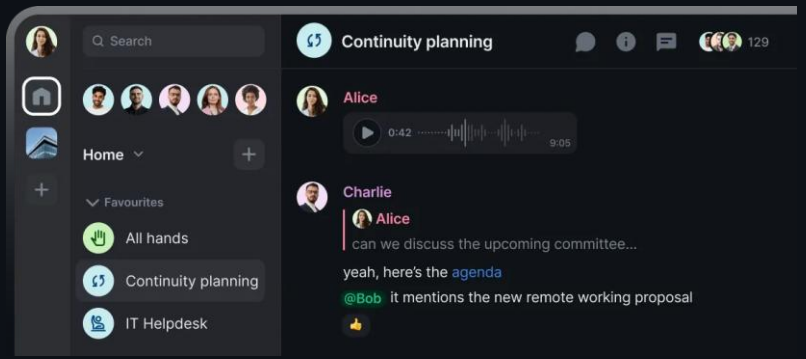
War story

War Story: Game Leaks, Hackers and Forums



MISSKEY

misskey-hub.net

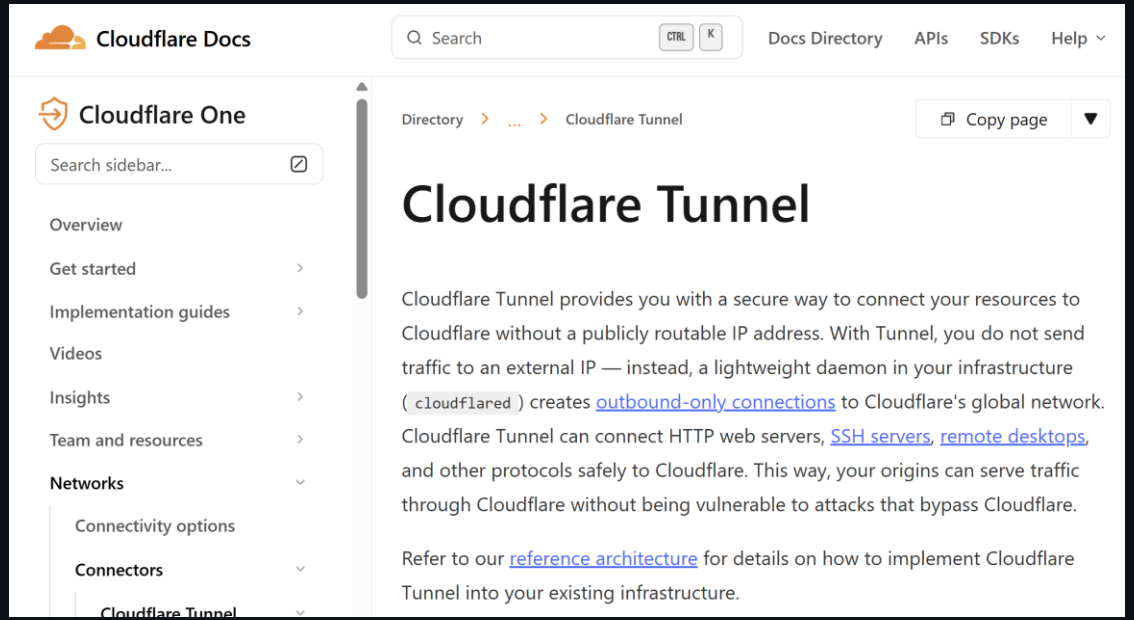


Element / Matrix

element.io

War story

War Story: Game Leaks, Hackers and Forums



https://developers.cloudflare.com/cloudflare-one/networks/connectors/cloudflare-tunnel/

BACKGROUND

Why Matrix?

The E2EE Forensic Challenge

- Signal, Telegram, WhatsApp & Wire widely adopted
- Encrypted comms shield criminal activities from detection
- Law enforcement capabilities severely hampered
- Existing research focuses on client-side artifacts only

Matrix: A Unique Target

Self-hostable
Organizations run their own Homeservers — full data control

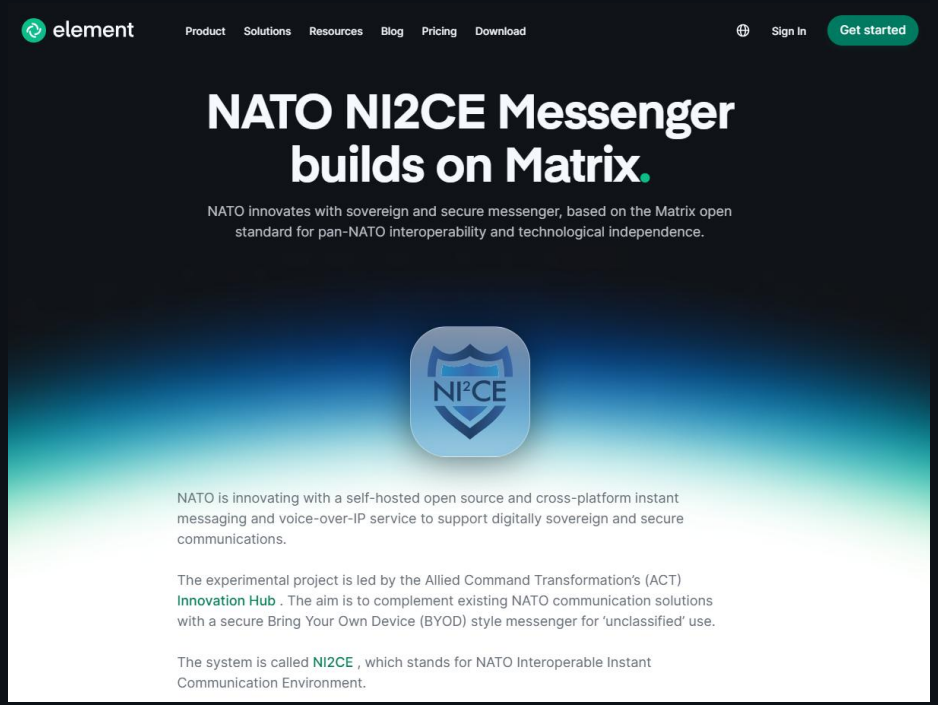
Federated
Decentralized rooms replicated across servers

Government Use
French govt & NATO adopted Matrix / Element

Research Gap
No prior study of Synapse server-side forensics

BACKGROUND

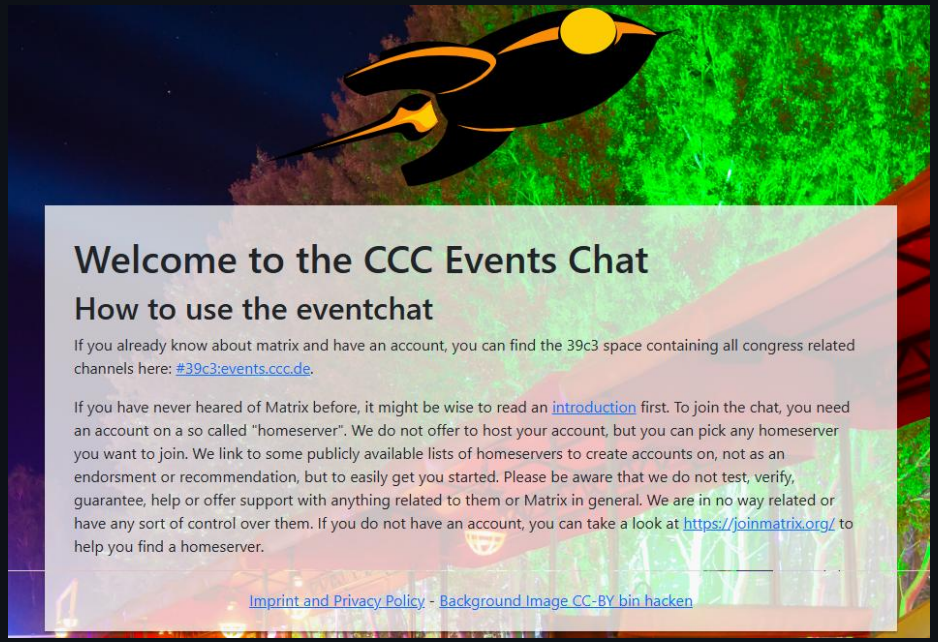
Global Usage | NATO



https://element.io/en/case-studies/nato

BACKGROUND

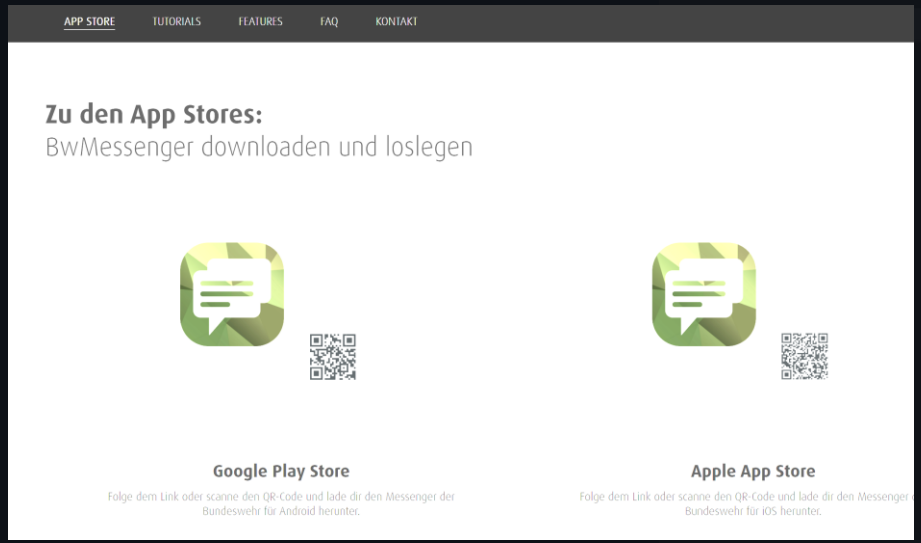
Global Usage | Chaos Computer Club



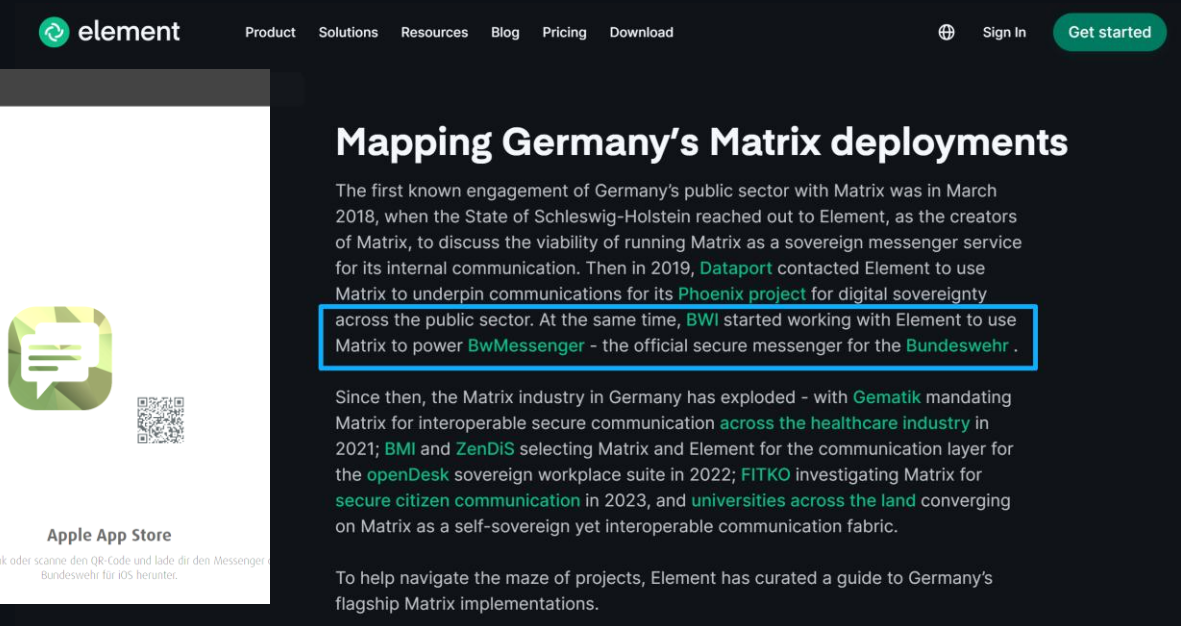
https://chat-info.events.ccc.de/

BACKGROUND

Global Usage | German Armed Forces



https://messenger.bwi.de/bwmessenger



https://element.io/en/matrix-in-germany

ARCHITECTURE

Matrix Protocol Architecture

Homeservers

- Manage server-side ops, user authentication & federation.
- Each runs Synapse — stores encrypted room replicas.

Client Apps

- Element (desktop/mobile/web).
- Provide user interfaces and handle local message encryption/decryption.

App Services

Protocol bridges connecting Matrix with external platforms (Slack, WeChat, etc.) via bot interfaces.

Four Communication Paradigms

1. Same Homeserver

Alice ↔ Emily (both on HS-A)

2. Cross-Server Federation

Bob (HS-B) ↔ Alice (HS-A)

3. Matrix ↔ Bridge

Emily (Matrix) ↔ WeChat user Wang

4. Cross-Platform via Matrix

WeChat ↔ Slack through federation

ENCRYPTION

Olm vs. Megolm: Encryption Protocols

Olm — Pairwise		Megolm — Group	
Security Model	Forward + Backward Secrecy	Security Model	Forward Secrecy only
Algorithm	Double Ratchet (Signal)	Algorithm	Unidirectional ratchet
Self-Healing	✓ YES	Self-Healing	✗ NO
Scalability	$O(n^2)$ key exchanges	Scalability	O(n) — efficient
Compromise Impact	Time-limited only	Compromise Impact	Full session exposed
Forensic Evidence	**High volume, distributed**	Forensic Evidence	**Low volume, centralized**
Primary Use	1-to-1 device sessions	Primary Use	Group chat rooms

OLM · PAIRWISE

Double Ratchet Algorithm

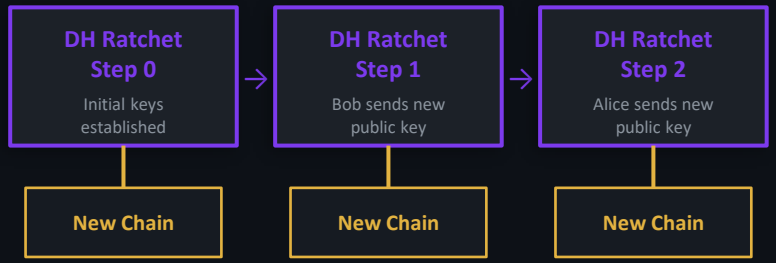
Forward secrecy + Post-compromise security — Two ratchets working in tandem

① Symmetric-Key Ratchet (Chain Key)

Message Key (MK) — encrypts one message, used once then discarded

→ Forward Secrecy: past messages safe even if current key is exposed

② Diffie-Hellman Ratchet (Root Key)



Each DH exchange = fresh entropy injected → new symmetric chain spawned

→ Post-Compromise Security: future messages safe after a past key leak

Combined Power

Two ratchets run simultaneously — symmetric ratchet for per-message keys, DH ratchet for periodic re-keying

Forward Secrecy

Each message key is derived fresh and deleted after use. Compromise of current key does not expose past messages

Post-Compromise Heal

After a breach, the next DH ratchet step injects new entropy. The session "self-heals" — future messages are re-secured

Cost: $O(n^2)$

Every pair of devices needs its own Olm session. For n devices: $O(n^2)$ sessions — only feasible for pairwise (1-to-1) use

https://signal.org/docs/specifications/doubleratchet/

ENCRYPTION

Olm vs. Megolm: Encryption Protocols

Olm — Pairwise		Megolm — Group	
Security Model	Forward + Backward Secrecy	Security Model	Forward Secrecy only
Algorithm	Double Ratchet (Signal)	Algorithm	Unidirectional ratchet
Self-Healing	✓ YES	Self-Healing	✗ NO
Scalability	$O(n^2)$ key exchanges	Scalability	O(n) — efficient
Compromise Impact	Time-limited only	Compromise Impact	Full session exposed
Forensic Evidence	**High volume, distributed**	Forensic Evidence	**Low volume, centralized**
Primary Use	1-to-1 device sessions	Primary Use	Group chat rooms

MEGOLM · GROUP

Unidirectional Ratchet Algorithm

One-way advancement — optimised for scalable group messaging at the cost of backward secrecy

Megolm Session: Single Forward-Only Ratchet Chain

Message Key (MK) — derived from each ratchet state, used to encrypt exactly one message

⚠ Cannot go backwards — once S_i advances to S_{i+1}, state S_i is permanently deleted

How it works

A single symmetric ratchet advances forward on each message. The session key is shared once via Olm to all room members.

Forward Secrecy

Past message keys are deleted after use. Compromising the current state does not decrypt earlier messages.

No Backward Secrecy

If state S_i is compromised, ALL future messages (S_{i+1}, S_{i+2} …) can be decrypted — until a key rotation event.

Cost: O(n)

One session per room, shared among all n members. Efficient regardless of group size — ideal for large group chats.

https://spec.matrix.org/v1.17/olm-megolm/megolm/

BACKGROUND

Matrix's encryption

Springer Nature Link
https://link.springer.com › article

Matrix protocol: a comprehensive systematic mapping study

by JAP Martins · 2026 — **Matrix** builds upon existing components to establish an open **protocol** that empowers users to have full control over their data and communication. Read more

[BOOK] **Matrix Protocol End**-to-**End Encryption**: The Complete Guide for Developers and Engineers

W Smith - 2025 - books.google.com

… In **summary**, native **end**-to-**end encryption** within **Matrix** is an indispensable design choice … By embedding E2EE at the **protocol** level, **Matrix** protects message confidentiality against a …

☆ Save Cite Related articles

BACKGROUND

Element & Matrix: Understanding the Relationship

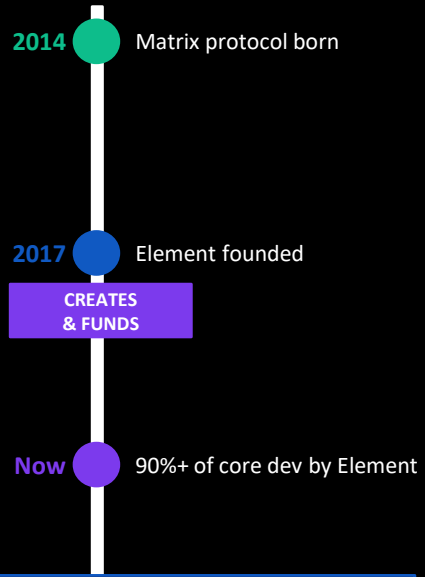
Matrix.org

Open Standard · Non-Profit

Decentralised, open protocol for real-time communications. Created in 2014 as the "missing comms layer of the open web."

The Matrix.org Foundation

- Non-profit community interest company
- Independently manages the Matrix specification
- 4 directors (2 from original founding team)
- Governing Board: elected ecosystem reps



Element

For-Profit Company · Founded 2017

Built by the same founding team to demonstrate what can be built on Matrix, and to attract investment for further Matrix development.

Contributions to Matrix

90%+ of key server components & SDKs

All contributions made free of charge to the Matrix project

Revenue Model

Enterprise subscriptions (Element Server Suite Pro) fund ongoing Matrix development

Open Source

AGPL v3 license — free for individuals & FOSS community; proprietary features for enterprise

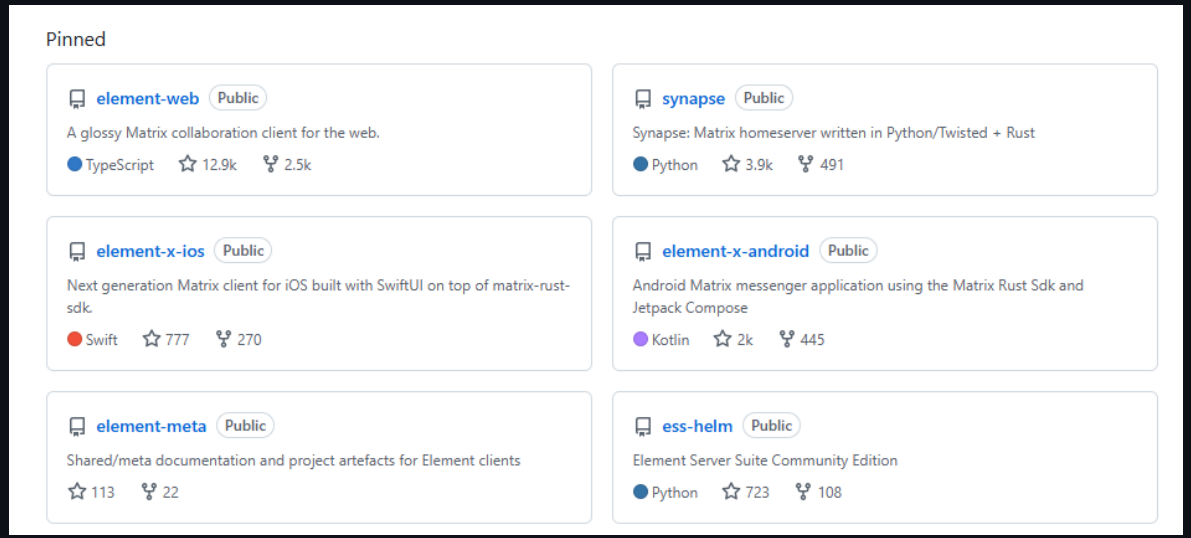
Foundation Support

Element manages matrix.org homeserver on behalf of The Matrix.org Foundation

https://element.io/en/matrix-relationship

BACKGROUND

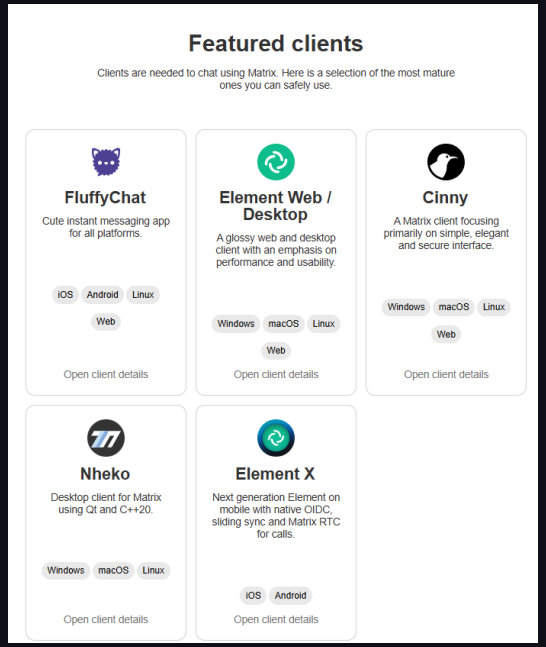
Element Github Repos

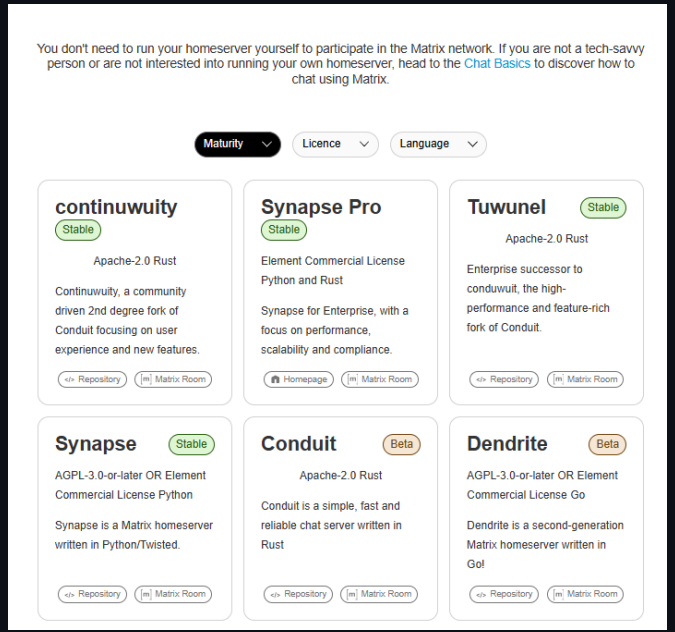


https://github.com/element-hq

BACKGROUND

Alternative Ecosystem | Clients & Servers





https://matrix.org/ecosystem/clients/

RESEARCH QUESTIONS

What Are We Trying to Find?

RQ1

What categories of forensic artifacts persist in Synapse homeserver deployments despite end-to-end encryption protections?

→ Authentication records · Device fingerprints · File transfer metadata · Cryptographic key events · Message metadata

RQ2

To what extent can investigators reconstruct communication timelines, user relationships, and behavioral patterns from server-side metadata alone?

→ Reconstruct timelines · Map social networks · Identify participants · Document file sharing activity

⭐ Contribution: First systematic server-side forensic study of Matrix · SynExtract open-source tool · Practical investigator guidance

METHODOLOGY

Experimental Setup & Test Scenarios

Server Environment

OS	Ubuntu 22.04 LTS (VM)
Resources	2 vCPU · 4 GB RAM
Synapse	v1.133.0 (Docker Compose)
Database	SQLite — homeserver.db
Admin UI	Synapse Admin v0.11.1
Clients	Element Desktop (Linux) + Element Android 1.6.44

Test Scenarios (3 User Accounts)

Account Setup	Registration and login
Direct Messaging	Text, emoji, formatted text
File Sharing	Image, document, video
Group Room	Create + join 3-person room
Group Messaging	Group text and media sharing
Voice & Video	Call initiation and termination

Two Analysis mode

Log only

Database only

LOG ARTIFACTS

Synapse Log File Artifacts

Log location: /var/lib/docker/containers/[id]/[id]-json.log

Authentication & Sessions

User ID · Device ID · IP · User-Agent · Platform · Login timestamps

Media File Transfers

Upload path · Media URI · File size · Upload/download user · Duration

Message Metadata

Room ID · Event ID · Sender · Timestamp (content encrypted)

VoIP Call Preparation

TURN server queries · VoIP capability checks · User ID · Client version

Message Receipts & Typing

Read receipts · Read timestamps · Typing indicators per room

Cryptographic Key Management

One-time key uploads · Device key updates · Cross-signing setup · Key backup ops

⚠ Message content remains cryptographically protected — only metadata is recoverable from logs

LOG ARTIFACTS

Authentication

Login

{"log":"2025-09-01 13:13:07,662 - synapse.handlers.auth - 1009 - INFO - POST-135 - Logging in user @root:matrix.redundan3y.com on device LNOMLDNWLQ\n","stream":"stderr","time":"2025-09-01T13:13:07.662949991Z"}

Device Registration

{"log":"2025-09-01 13:11:01,156 - synapse.access.http.8008 - 515 - INFO - GET-115 - 183.194.74.162 - 8008 - {None} Processed request: 0.000sec/0.000sec (0.000sec, 0.000sec) (0.000sec/0.000sec/0) 78B 200 \"GET /_matrix/client/v3/login HTTP/1.1\" \"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36\" [0

LOG ARTIFACTS

Message Activity

Encrypted Message Send

{@wushufan:matrix.redundan3y.com} Processed request: 0.036sec/0.001sec (0.005sec, 0.000sec) (0.003sec/0.006sec/13) 59B 200 \"PUT /_matrix/client/r0/rooms/!cPjsVTKCjDgQlGNQgF:matrix.redundan3y.com/send/m.room.encrypted/$local.66232120-5f23-4857-a2f2-40dcffed9418 HTTP/1.1\" \"Element/1.6.44 (HUAWEI ALN-AL00; Android 12; ALN-AL00 4.2.0.183(C00E165R4P16); Flavour GooglePlay; MatrixAndroidSdk2 1.6.44)\" [0 dbevts]\n","stream":"stderr","time":"2025-09-01T14:05:46.724332338Z"}

Message Receipt Confirm

{@wangyikai:matrix.redundan3y.com} Processed request: 0.010sec/0.000sec (0.003sec, 0.000sec) (0.001sec/0.003sec/6) 2B 200 \"POST /_matrix/client/r0/rooms/!cPjsVTKCjDgQlGNQgF:matrix.redundan3y.com/receipt/m.read/$ijSUhoIaxvEDW0TPepfF49pVoHQptxxHpgxRX5n0K54 HTTP/1.1\" \"Element/1.6.44 (Xiaomi 22101317C; Android 14; UKQ1.230917.001 release-keys; Flavour GooglePlay; MatrixAndroidSdk2 1.6.44)\" [0 dbevts]\n","stream":"stderr","time":"2025-09-01T14:05:47.526403245Z"}

LOG ARTIFACTS

Message Activity

Typing Indicators

{@wangyikai:matrix.redundan3y.com} Processed request: 0.004sec/0.000sec (0.000sec, 0.000sec) (0.000sec/0.000sec/1) 2B 200 \"PUT /_matrix/client/r0/rooms/!cPjsVTKCjDgQlGNQgF:matrix.redundan3y.com/typing/@wangyikai:matrix.redundan3y.com HTTP/1.1\" \"Element/1.6.44 (Xiaomi 22101317C; Android 14; UKQ1.230917.001 release-keys; Flavour GooglePlay; MatrixAndroidSdk2 1.6.44)\" [0 dbevts]\n","stream":"stderr","time":"2025-09-01T14:05:48.228264182Z"}

LOG ARTIFACTS

Media File Transfer Operations

File Upload

{"log":"2025-09-01 14:08:44,070 - synapse.media.media_repository - 334 - INFO - POST-1817 - Stored local media in file '/data/media_store/local_content/xz/Ku/KSXBXiYuyEUPmZNCMJTI'\n","stream":"stderr","time":"2025-09-01T14:08:44.070412461Z"}
{"log":"2025-09-01 14:08:44,074 - synapse.rest.media.upload_resource - 131 - INFO - POST-1817 - Uploaded content with URI 'mxc://matrix.redundan3y.com/xzKuKSXBXiYuyEUPmZNCMJTI'\n","stream":"stderr","time":"2025-09-01T14:08:44.074509698Z"}
{"log":"2025-09-01 14:08:44,075 - synapse.access.http.8008 - 515 - INFO - POST-1817 - 183.192.83.35 - 8008 - {@wangyikai:matrix.redundan3y.com} Processed request: 0.006sec/0.000sec (0.001sec, 0.000sec) (0.000sec/0.004sec/2) 70B 200 \"POST /_matrix/media/r0/upload HTTP/1.1\" \"Element/1.6.44 (Xiaomi 22101317C; Android 14; UKQ1.230917.001 release-keys; Flavour GooglePlay; MatrixAndroidSdk2 1.6.44)\" [0 dbevts]\n","stream":"stderr","time":"2025-09-01T14:08:44.075206412Z"}

LOG ARTIFACTS

Media File Transfer Operations

File Download

{@wushufan:matrix.redundan3y.com} Processed request: 0.006sec/-0.000sec (0.001sec, 0.000sec) (0.001sec/0.000sec/1) 118383B 200 \"GET /_matrix/client/v1/media/download/matrix.redundan3y.com/xzKuKSXBXiYuyEUPmZNCMJTI HTTP/1.1\" \"Element/1.6.44 (HUAWEI ALN-AL00; Android 12; ALN-AL00 4.2.0.183(C00E165R4P16); Flavour GooglePlay; MatrixAndroidSdk2 1.6.44)\" [0 dbevts]\n","stream":"stderr","time":"2025-09-01T14:08:44.801976355Z"}

LOG ARTIFACTS

Cryptographic Key Management

One-Time Key Upload

{"log":"2025-09-13 06:53:42,601 - synapse.handlers.e2e_keys - 951 - INFO - POST-70566 - Adding one_time_keys dict_keys(['signed_curve25519:AAAAAAAAADI']) for device 'VMKJFLDDHH' for user '@zhangxuepei:matrix.redundan3y.com' at 1757746422590\n","stream":"stderr","time":"2025-09-13T06:53:42.601604278Z"}

Device Key Update

{"log":"2025-09-13 07:00:25,468 - synapse.handlers.e2e_keys - 820 - INFO - POST-70955 - Claimed one-time-keys: signed_curve25519:AAAAAAAAAA4 for @zhangxuepei:matrix.redundan3y.com:VMKJFLDDHH\n","stream":"stderr","time":"2025-09-13T07:00:25.468716157Z"}

LOG ARTIFACTS

Cryptographic Key Management

Cross-Signing Setup

{@wangyikai:matrix.redundan3y.com} Processed request: 0.008sec/0.001sec (0.001sec, 0.001sec) (0.000sec/0.004sec/3) 2B 200 \"PUT /_matrix/client/r0/user/@wangyikai:matrix.redundan3y.com/account_data/m.secret_storage.key.45b4fc66-6599-4973-b04d-bf7b2f0255f7 HTTP/1.1\" \"Element/1.6.44 (Xiaomi 22101317C; Android 14; UKQ1.230917.001 release-keys; Flavour GooglePlay; MatrixAndroidSdk2 1.6.44)\" [0 dbevts]\n","stream":"stderr","time":"2025-09-10T00:07:11.458578407Z"}

Key Backup Operations

{@zhangxuepei:matrix.redundan3y.com} Processed request: 0.005sec/0.000sec (0.001sec, 0.000sec) (0.001sec/0.002sec/5) 22B 200 \"PUT /_matrix/client/v3/room_keys/keys?version=1 HTTP/1.1\" \"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Element/1.11.111 Chrome/138.0.7204.224 Electron/37.3.0 Safari/537.36\" [0 dbevts]\n","stream":"stderr","time":"2025-09-16T10:50:05.811396691Z"}

DATABASE ARTIFACTS

Synapse Database: 175 Tables

175	6	SQLite
DB Tables	Artifact Categories	Default Format

User Identification

user_ips + devices tables: User ID, Device ID, Device Name, IP Address, User-Agent — links digital identity to physical person

User Relationships

room_memberships, users_who_share_private_rooms: Bidirectional contact mapping; member count distinguishes DM (2) vs group (3+)

Message Content

Non-E2EE rooms: event_search_content table stores plaintext. Event relations reconstruct threads; reactions in aggregation_key

Message Metadata

events table: Event ID, message type, Room ID, Device metadata in internal_metadata — provenance without decryption

Read Receipts

receipts_linearized: Read user, event ID, millisecond-resolution timestamps — behavioral timeline reconstruction

Media Transfers

media_repository: Media ID, file type, file size, uploader, created time, last access — complete file sharing record

DATABASE ARTIFACTS

User Identification Information

Source Tables: **user_ips**

user_id	device_id	ip	user_agent (abbreviated)	last_seen
@zhangxuepei:matrix.test.com	VMKJFLDDHH	114.XX.XX.205	Element/1.6.44 · HONOR PGT-AN10 · Android 13	2025-09-12 16:13
@test:matrix.test.com	BXQYWKGTNU	183.XX.XX.118	Element/1.11.111 · Linux x86_64 · Chrome/138	2025-09-12 16:40
@wangyikai:matrix.test.com	JOGKWUVOGV	203.XX.XX.252	Element/1.6.44 · Xiaomi 22101317C · Android 14	2025-07-01 08:19
@wangyikai:matrix.test.com	JOGKWUVOGV	183.XX.XX.35	Element/1.6.44 · Xiaomi 22101317C · Android 14	2025-07-01 09:01
@wangyikai:matrix.test.com	JOGKWUVOGV	162.XX.XX.218	Element/1.6.44 · Xiaomi 22101317C · Android 14	2025-07-01 08:19

What Investigators Can Recover

① Real Identity

user_id links a Matrix handle to a physical person. Combined with device_id, investigators can tie every action to a specific account and device.

② IP Address History

@wangyikai accessed the server from 3 different IPs on the same device — revealing location changes and potential VPN/proxy usage (162.158.x = Cloudflare).

③ Device Fingerprint

user_agent encodes OS, device model, and client version. All three @wangyikai rows share device JOGKWUVOGV and identical UA — confirming one physical device, multiple sessions.

DATABASE ARTIFACTS

User Identification Information

Source Table: `devices`

user_id	device_id	display_name	last_seen · ip	ua	hid
@wangyikai:…	WkrIAjG6Jntx…TNo	master signing key	–	–	1
@wangyikai:…	of6h9JqEfurQ…Bnk	self_signing signing key	–	–	1
@wangyikai:…	cxcReWfu…xn8	user_signing signing key	–	–	1
@wushufan:…	KLCVGFXBUJ	Element Android	2025-09-12 183.XX.XX.254	HUAWEI ALN-AL00 · Android 12	0
@wushufan:…	WPXa2OT+lZX8…rU	master signing key	–	–	1
@wushufan:…	tbtBCsNAQ7fY…Lns	self_signing signing key	–	–	1
@wushufan:…	7RaPxMwkmSg0…ew	user_signing signing key	–	–	1

What Investigators Can Recover

① Real vs. Cryptographic Device IDs

hidden=0 rows are physical devices with a short alphanumeric device_id, IP address, and user-agent. hidden=1 rows are cross-signing keys masquerading as devices — their long base64 IDs are Ed25519 public keys, not hardware.

② Cross-Signing Key Enumeration

Each user exposes all three cross-signing key types: master signing key (MSK), self_signing key (SSK), and user_signing key (USK). Their presence confirms the user activated verified device cross-signing and anchors the trust chain.

③ Device Fingerprint + Session Anchor

KLCVGFXBUJ belongs to a HUAWEI ALN-AL00 (Android 12) last seen at 183.XX.XX.254. The display_name 'Element Android' was set by the user — names are investigator-readable labels that can corroborate testimony.

Physical device (hidden=0) · @wangyikai cross-signing keys (hidden=1) · @wushufan cross-signing keys (hidden=1) · hidden=1 (key-only, not a real device)

DATABASE ARTIFACTS

User Relationships

Source Table: `users_who_share_private_rooms`

user_id	other_user_id	room_id
@wushufan	@wangyikai	!cPjsVTKC…
@wangyikai	@wushufan	!cPjsVTKC…
@zhangxuepei	@wangyikai	!sAaSjzzR…

`room_memberships`

user_id	sender	membership	display_name
@zhangxuepei	@wangyikai	invite	十四
@zhangxuepei	@zhangxuepei	join	十四
@wangyikai	@wangyikai	join	wang

Direct Message (@wushufan ↔ @wangyikai)

Private Room (@zhangxuepei ↔ @wangyikai)

💡 DM vs. Group Detection: room with 2 members = Direct Message · 3+ members = Group Room
Each bidirectional pair (A→B + B→A) confirms a confirmed mutual contact relationship.

Invite Chain Detected

Row 1: sender ≠ user_id → @wangyikai invited @zhangxuepei (establishes social link)
Row 2: sender = user_id → @zhangxuepei accepted the invite

What Investigators Can Recover

① Social Graph Mapping

users_who_share_private_rooms exposes the complete contact network: every user who ever communicated privately is listed, even if messages are E2EE-protected.

② Invite Chain = Relationship Proof

When sender ≠ user_id in room_memberships, an explicit social link is proven — @wangyikai recruited @zhangxuepei into the room, independent of message content.

③ Display Name Persistence

display_name (e.g. '十四', 'wang') persists in the database and may correspond to real-world identifiers, nicknames, or codenames used across multiple rooms.

DATABASE ARTIFACTS

Group Room Message Analysis · events

Room: !cPjsVTKCjDgQlGNQgF · Source Table: events + events_json

depth	event type	sender	content (from events_json)	origin_server_ts	
1	m.room.create	@wangyikai	creator: "wang" · room_version: 10	01 Aug 08:05:21	*Room created*
2	m.room.member (join)	@wangyikai	displayname: "wang" · membership: join	08:05:21	
3	m.room.power_levels	@wangyikai	wangyikai: 100 · wushufan: 100	08:05:21	*Admin config*
7	m.room.member (invite)	@wangyikai	invite → @wushufan · is_direct: true	08:05:22	*Invite*
8	m.room.message	@wangyikai	"hello" · device: JOGKWUVOGV	08:05:22	
9	m.room.member (join)	@wushufan	membership: join · device: KLCVGFXBUJ	08:13:32	*Wu joins*
10	m.room.message	@wushufan	"hihi" · device: KLCVGFXBUJ	08:13:43	
11	m.room.message	@wangyikai	"nice" · txn_id present	08:14:12	
12	m.room.message	@wangyikai	"这里就随便聊天了"	08:14:22	*Plaintext*
13	m.room.message	@wushufan	"一定要用你配的服务器吗"	08:14:44	*Plaintext*
15	m.room.message	@wangyikai	“对的"	08:15:55	*Sensitive*

events table reveals: complete room lifecycle (create → invite → join → message) with sender, device_id, depth ordering, and millisecond timestamps — **plaintext content recoverable in non-E2EE rooms.**

DATABASE ARTIFACTS

Group Room Message Analysis · event_relations

Source Table: event_relations — maps reactions, edits, and thread replies to parent events

m.annotation

Reactions

👍 / 👎 / 🧐 / 🤗 / 👇

child event_id	aggregation_key
$e0Mw…	👎
$XW_i…	🧐
$nnoJ…	🧐
$2l9d…	🤗
$AyiI…	👇

5 emoji reactions on message $Bv_Dd6d… — reveals emotional responses, social dynamics, and confirms users actively read and engaged with the message.

m.replace

Message Edit

🖍️ Edit recorded

child event_id	aggregation_key
$MXe4…	—

$MXe4… replaced $o6Qj… — the edit event links new text to original. Both versions persist on server; investigators can recover original + edited content.

m.reference

Thread Replies

🧵 3 replies

child event_id	aggregation_key
$b16C…	—
$jD6D…	—
$WJe-…	—

3 events reference $1RyBkfv… as thread root, revealing a sub-conversation. Thread structure reconstructs reply chains even when message content is encrypted.

event_relations exposes social context invisible to basic log analysis: who reacted and how (**m.annotation**), what was edited and when (**m.replace**), and which messages sparked follow-up threads (**m.reference**).

DATABASE ARTIFACTS

Read Receipt Artifacts

Source Table: `receipts_linearized`

stream_id	room_id (abbreviated)	user_id	event_id (abbreviated)	timestamp (ms)
103	!cPjsVTK…com	@wangyikai	$8UE8WBO…Ac	1757662142984
104	!cPjsVTK…com	@wushufan	$Njv4j3_…gw	1757662754235
117	!tShmdQh…com	@wangyikai	$Pf0rh-N…ZE	1757746830236
116	!tShmdQh…com	@zhangxuepei	$RuY5FeW…SQ	1757746819951
167	!sAaSjzz…com	@wangyikai	$C-ReYxD…HE	1757760314394

Read-Order Timeline (stream_id = global ordering)

What Investigators Can Recover

① Who Read What, Exactly

Each row links user → event_id → room with millisecond timestamp. Investigators know exactly which message each user last read in every room.

② Global Read Order via stream_id

stream_id is monotonically increasing. Rows 116 → 117: @zhangxuepei read their message 10 seconds BEFORE @wangyikai — both in the same room.

③ Cross-Room Activity Mapping

@wangyikai appears in all 3 rooms (sid 103, 117, 167), revealing their active communication rooms without any message content.

KEY FINDINGS

What Can Investigators Recover?

✓ Recoverable Despite E2EE

- Real identity: User IDs, IP addresses, device fingerprints
- Social graph: Who communicated with whom and when
- Communication timeline: Send/read timestamps per message
- File transfer records: What was shared, by whom, file size
- Typing and reading behavior: Real-time interaction patterns
- Authentication history: Login times, devices, IP changes
- Cryptographic activity: Key uploads, cross-signing events
- Plaintext content: Rooms without E2EE enabled

✗ Protected by Encryption

- Message text content (in E2EE rooms)
- Media file contents
- Voice / video call audio and video
- Reaction content (in E2EE rooms)

Key Insight

Even with perfect E2EE, server-side metadata reveals who, when, where, and how often — sufficient to build a complete investigative picture.

DISCUSSION

Discussion, Limitations & Future Work

Contributions

✓ First systematic forensic study of Synapse server-side artifacts

✓ SynExtract — open-source Python tool for automated extraction & reporting

✓ Identified recoverable evidence across 175 DB tables and structured logs

✓ Practical guidance for law enforcement with lawful server access

Limitations

! Restricted to single-server deployments — federated artifacts not examined

! No active-session cryptographic key interception explored

! Element client forensics (desktop / mobile / web) not covered

Future Work

(1) Message interception during active sessions with admin privileges | (2) Systematic client-side forensics across all Element platforms

Conclusion

End-to-End Encryption ≠ Forensic Invisibility

Substantial metadata persists on Synapse servers despite strong E2EE — enabling identity, timeline, and relationship reconstruction

175 database tables + structured logs provide rich forensic artifacts accessible to investigators with lawful server access

SynExtract automates extraction, correlation, and visualization — reducing analyst effort significantly

Matrix's self-hosting model creates unique forensic opportunities absent in cloud-based messaging platforms

github.com/redundan3y/SynExtract
redundan3y@protonmail.com