Forensic Science International: Digital Investigation 56 (2026) 302049

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

DFRWS EU 2026 - Selected Papers from the 13th Annual Digital Forensics Research Conference Europe

DFRWS down the Rabbit-Hole: A forensic analysis of the Matrix protocol and Synapse server

Yikai Wang [a,b,*], Xuepei Zhang [a], Shufan Wu [a], Yan Chen [a,**]

[a] Department of Criminal Justice, East China University of Political Science and Law, 1575 Wanhangdu Road, Changning District, Shanghai, 200042, China
[b] Shanghai Jueyin Digital Forensic Institute, No. 1247 Meichuan Road, Putuo District, Shanghai, 200333, China

ARTICLE INFO

Keywords:
Digital forensic
Matrix protocol
Synapse server
End-to-end encryption

ABSTRACT

The widespread adoption of end-to-end encrypted messaging platforms presents significant challenges for digital forensic investigations. This paper presents the first comprehensive forensic analysis of Synapse, the official Matrix Homeserver implementation, focusing on server-side artifacts persisting in both database structures and system logs despite end-to-end encryption. Through systematic examination of production deployments, we identify recoverable digital evidence across 175 database tables and structured log entries, including authentication records, communication timelines, device fingerprints, and file transfer metadata. While message content remains cryptographically protected, our analysis demonstrates substantial investigative value in metadata accessible to investigators with lawful server access. We developed SynExtract, a specialized tool that automates extraction and correlation of artifacts from both Synapse databases and log files. Our findings provide practical guidance and a tool for law enforcement personnel conducting forensic examinations of Matrix infrastructure in criminal investigations.

1. Introduction

The growing public concern over digital privacy has fueled widespread adoption of end-to-end encrypted (E2EE) messaging applications (Christy, 2022). Following numerous data breaches and surveillance revelations, users increasingly prioritize the confidentiality of their digital communications (Global trends & forecast). This trend has propelled applications such as Signal, Telegram, WhatsApp, and Wire to prominence, as they implement robust E2EE protocols that safeguard conversation content from unauthorized access.

However, these end-to-end encrypted messaging platforms, while serving legitimate privacy needs, have inadvertently provided criminal actors with highly secure communication channels that can shield illicit activities from detection (Pisaric, 2022). Law enforcement agencies worldwide have expressed concerns about the challenges these technologies present to their investigative capabilities (Hartel and van Wegberg, 2023). Consequently, digital forensic research focusing on these applications has become increasingly crucial. Several studies, including those by Jaeckel et al. (2025) and Onik et al. (2025), have explored methods for extracting and analyzing digital evidence from popular E2EE applications, though significant challenges remain. Among the emerging technologies in this space, the Matrix protocol stands out as a newer entrant with distinctive self-hosting capabilities, allowing organizations to maintain complete control over their communication infrastructure. This feature has made it particularly attractive to government institutions, such as the French government and the NATO (Element) ("Defence | Military |C3).

2. Related work

Although much research has explored encrypted messaging apps like Signal, WhatsApp, Telegram, and iMessage, little attention has been paid to the Matrix protocol and its self-deployable server application, Synapse.

The forensic examination of Matrix communications presents unique challenges compared to traditional messaging platforms. In Schipper et al. (2021), the authors conducted one of the first comprehensive forensic analyses of the Matrix protocol and Riot.im application, concentrating on message artifacts and storage mechanisms. Their research reveals that files maintain original properties during

* Corresponding author. Department of Criminal Justice, East China University of Political Science and Law, 1575 Wanhangdu Road, Changning District, Shanghai, 200042, China.
** Corresponding author.
E-mail addresses: redundan3y@protonmail.com (Y. Wang), chengyan@ecupl.edu.cn (Y. Chen).

https://doi.org/10.1016/j.fsidi.2026.302049

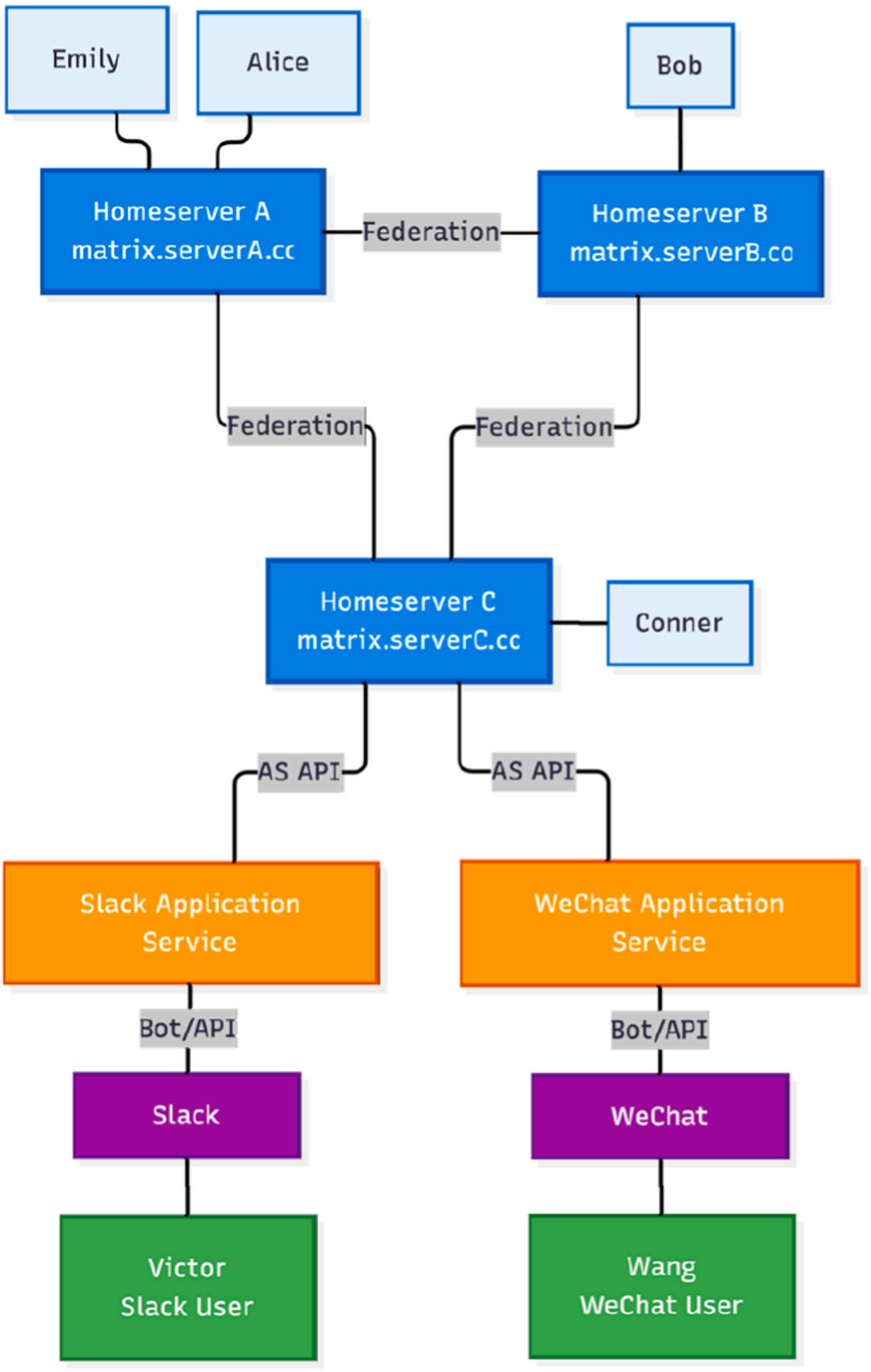

Fig. 1. Matrix ecosystem architecture demonstrating federated network topology with cross-platform bridging capabilities.

transmission, timestamps are server-generated, and valuable data exists in a LevelDB database and an encrypted SQLite database. However, few studies have since been conducted on the new generation Matrix client, Element.

From a security evaluation perspective, the Matrix protocol's infrastructure has undergone systematic vulnerability assessments. Stenhav (Stenhav) conducts a comprehensive security evaluation of the Matrix protocol's Server–Server API, focusing on identifying and analyzing potential vulnerabilities through a systematic threat modelling and pentesting approach. Employing STRIDE and DREAD frameworks, this research identified 16 critical vulnerabilities. The evaluation revealed that the Server–Server API maintains secure, as the tested vulnerabilities proved resistant to exploitation, though the identification of untested vulnerabilities highlighted ongoing security enhancement requirements.

Complementing these infrastructure assessments, Albrecht et al. (Albrecht et al.) identified several practically exploitable cryptographic vulnerabilities in the Matrix protocol and Element client that undermine confidentiality and authentication guarantees against malicious servers. Their analysis revealed critical flaws including confidentiality breaks from Homeserver control over room membership and device lists, attacks on out-of-band verification through identifier confusion, impersonation attacks from protocol confusion and implementation bugs in

Megolm and Olm protocols, and theoretical IND-CCA breaks in encryption schemes for key backups and secret sharing. The study highlights the absence of formal cryptographic analysis in Matrix and the need for systematic security evaluations in federated communication systems.

3. Research questions and contributions

3.1. Research question

This research addresses two fundamental questions regarding Matrix server-side forensics:

RQ1: What categories of forensic artifacts persist in Synapse homeserver deployments despite end-to-end encryption protections?
RQ2: To what extent can investigators reconstruct communication timelines, user relationships, and behavioral patterns from server-side metadata alone?

Our primary objective is to systematically identify, categorize, and evaluate the forensic value of recoverable artifacts from Synapse Homeserver logs and databases. This includes developing automated extraction methodologies and assessing the investigative utility of metadata that persists independently of message content.

3.2. Contribution

This paper presents a forensic analysis of the Matrix protocol emphasizing server-side artifacts generated by Synapse, the official Homeserver implementation. Through empirical analysis of production server deployments, we identify and categorize recoverable digital evidence including authentication records, message metadata, file transfer logs, and cryptographic key exchange events accessible to investigators with appropriate legal authority. While cryptographic implementation prevents content inspection, our findings demonstrate that substantial forensically relevant metadata persists within server-side artifacts, enabling reconstruction of communication timelines, identification of participating parties, and analysis of file sharing activities. We developed SynExtract, an open-source forensic analysis tool that provides automated extraction and analysis capabilities for server-side artifacts. (“redundan3y/SynExtract) This research addresses a critical gap by providing the first systematic examination of server-side forensic artifacts within Matrix deployments, offering essential guidance for investigators while acknowledging fundamental limitations imposed by end-to-end encryption architecture.

4. Background on Matrix protocol and Element

4.1. Matrix protocol architecture

Matrix represents a sophisticated E2EE messaging protocol distinguished by its integration of cryptographic security mechanisms with self-deployable decentralized federation capabilities, and cross-platform bridging functionality. These architectural innovations position Matrix as a distinctive solution in contemporary secure communication systems.

Federation constitutes a core architectural principle that enhances network resilience through distributed message storage and processing. Each conversation room maintains encrypted replicas across multiple participating servers within the federation network.

The Matrix infrastructure operates through three interconnected components: Homeservers manage server-side operations and user authentication, Client applications provide user interfaces and local message handling, and Application Services establish protocol bridges connecting Matrix with external messaging ecosystems including third-party platforms such as Slack and WeChat.

As illustrated in Fig. 1, the given Matrix ecosystem demonstrates a federated network topology consisting of three interconnected

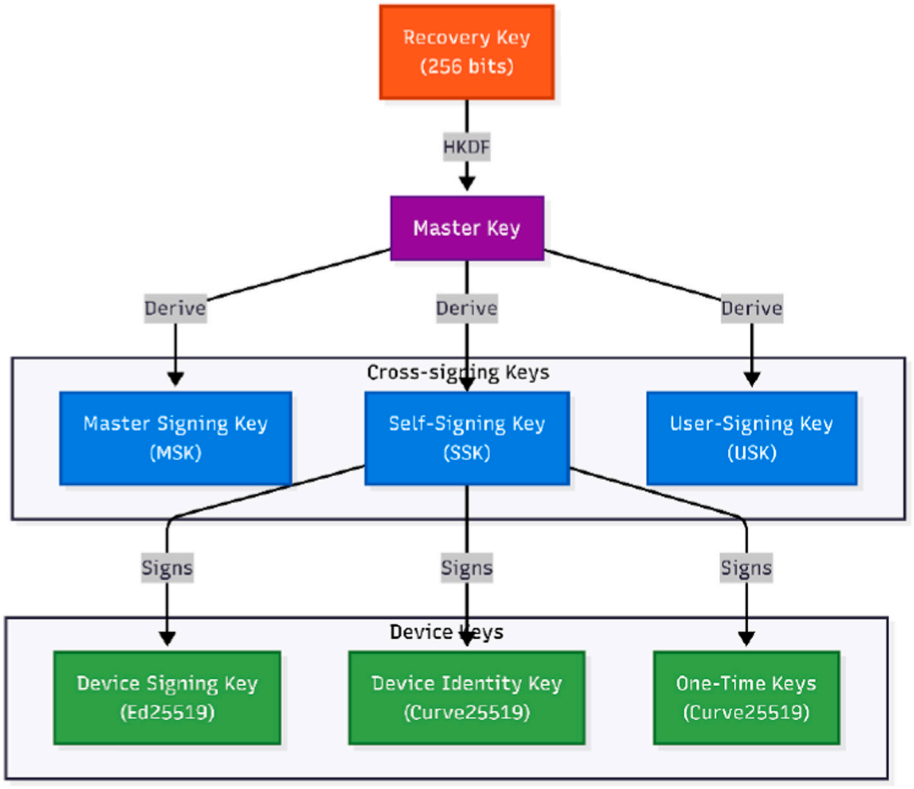

Fig. 2. Matrix recovery key derivation chain.

Homeservers (A, B, and C) with integrated Application Service bridging on *Homeserver C*. The federation connections enable seamless communication across all Matrix servers, while *Homeserver C* hosts Application Services that bridge to WeChat and Slack platforms through their respective APIs and bot interfaces. This architectural configuration supports multiple communication modalities across the network.

Within the Matrix federation, users Alice and Emily connect to *Homeserver A*, Bob connects to Homeserver B, and Conner connects to Homeserver C. External platform users Wang and Victor maintain their native connections to WeChat and Slack platforms respectively. The Application Services function as intermediary components, translating messages between Matrix's protocol and the external platforms' chatroom.

This architecture enables four distinct communication paradigms:

- Direct communication between users sharing the same Homeserver, such as Alice and Emily on *Homeserver A*;
- Cross-server messaging within the Matrix federation, exemplified by Bob communicating with Alice across different Homeservers;
- Hybrid communication linking Matrix users with external platform users through Application Service bridges, enabling Emily to exchange messages with WeChat user Wang;
- Transparent inter-platform communication, allowing Wang and Victor to communicate across WeChat and Slack platforms through the Matrix federation network as the intermediary layer.

The federation topology ensures that messages traverse through the Matrix network infrastructure, maintaining encryption and enabling cross-platform interoperability while preserving the native user experience on each platform. This design demonstrates Matrix's potential as a unifying communication substrate, enabling seamless interoperability across heterogeneous messaging platforms while preserving cryptographic security and maintaining distributed system reliability throughout all operational modes.

4.2. Encryption in Matrix

This section provides an overview of Matrix's encryption mechanisms necessary for understanding the forensic analysis presented in subsequent chapters. As the primary focus of this research is server-level forensics, the cryptographic details are presented concisely to establish the foundation for practical forensic techniques rather than providing comprehensive cryptographic analysis.

4.2.1. Olm & Megolm encryption systems

Matrix implements a device-based end-to-end encryption system using two complementary protocols: Olm for pairwise communication and Megolm for group messaging ("End-to-end encryption | Matrix Client). The encryption architecture is designed to operate across Matrix's federated infrastructure while maintaining cryptographic security independent of server trust.

Table 1
Comparison of Olm and Megolm protocol characteristics.

Characteristic	Olm	Megolm
Security Model	Forward + Backward	Forward only
Self-healing	YES	NO
Scalability	$O(n^2)$	$O(n)$
Session Setup	3DH key agreement	Key distribution
Compromise Impact	Time-limited	Full session
Forensic Evidence	High volume	Low volume
	Distributed	Centralized
Primary Use	Pairwise	Group chat
Storage Model	Complex multi-session	Simple session

4.2.2. Device-based encryption Model and recovery keys

Matrix's encryption operates on individual devices rather than user accounts. Each client login or installation generates unique cryptographic keys that establish device identity independent of Homeserver infrastructure. The system implements a hierarchical key derivation scheme rooted in a recovery key.

Matrix generates a recovery key from 256 bits of cryptographically secure entropy, encoded as a 58-character Base58 string following the format: *xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx*. This recovery key serves as the root entropy source for the entire cryptographic hierarchy.

As illustrated in Fig. 2, the key derivation hierarchy follows the following process: Recovery Key (256 bits) → Master Key → Cross-signing Keys → Device Keys.

The cryptographic hierarchy begins with a 256-bit recovery key that serves as the root entropy source for all subsequent key derivation. The recovery key undergoes HKDF-SHA-256 processing, a cryptographic key derivation function that extracts and expands entropy using HMAC-based operations, to generate a master key, which then deterministically derives the three cross-signing keys: Master Signing Key (MSK), Self-Signing Key (SSK), and User-Signing Key (USK). The Self-Signing Key subsequently signs and authenticates individual device keys, including the Ed25519 device signing key, Curve25519 device identity key, and multiple Curve25519 one-time keys.

Each device maintains three primary key types derived from this hierarchy: Ed25519 signing keys (32-byte) for device authentication and message signing, Curve25519 identity keys (32-byte) for ECDH key agreement during session establishment, and Curve25519 one-time keys for achieving perfect forward secrecy in session initialization (Li et al., 2023).

4.2.3. Olm protocol implementation

Olm handles pairwise encryption between devices using the Double Ratchet algorithm. Session establishment requires exchanging identity keys, signed prekeys, and consuming one-time keys from the target device. This process involves key request messages, session establishment events, and encrypted to-device messages (Albrecht et al., 2024).

The Double Ratchet maintains forward secrecy by deriving unique message keys for each communication. Each Olm session stores cryptographic state including root keys, chain keys, message keys, and Diffie-Hellman key pairs. These session objects are stored persistently in client databases and memory.

4.2.4. Megolm group encryption

Megolm provides scalable group messaging through symmetric encryption with session keys distributed via Olm channels. Each group conversation uses a Megolm session containing a ratchet state, message counter, and Ed25519 signing keys. Session keys are shared among authorized participants through encrypted to-device messages (Albrecht et al., 2024).

Megolm sessions involve session creation and key distribution events, ratchet advancement creating temporal message ordering, session sharing for new group members, and key rotation events triggered by membership changes. The hierarchical key derivation enables partial message recovery based on available session state. If a Megolm session at counter position i is available, all messages from position i onward can be decrypted, while earlier messages remain protected by forward secrecy.

4.2.5. Comparison between Olm and Megolm protocol

The fundamental architectural distinction between Olm and Megolm protocols lies in their approach to cryptographic security versus operational efficiency. Olm implements Signal's Double Ratchet algorithm optimized for pairwise communication with bidirectional security guarantees, while Megolm employs a simplified unidirectional ratchet designed for scalable group messaging.

Table 2
Experimental test scenarios.

Scenario Type	Participants	Operations
Account Setup	User A, B, C	Registration and login
Direct Messaging	User C User A, B	Text messages, emoji, formatted text exchange
File Sharing	User C User A, B	Image, document, video transfer
Group Creation	User C Users A, B	Create 3-person group room Join group room
Group Messaging	All 3 users	Group text conversations and media sharing
Voice Calls	User C User A	Voice call initiation and termination
Video Calls	User C User A	Video call initiation and termination

Olm provides comprehensive security through its dual-ratchet mechanism, combining symmetric key ratcheting with Diffie-Hellman ratcheting to achieve both forward secrecy and post-compromise security. The Diffie-Hellman ratchet enables healing properties that restore security after key material disclosure by introducing fresh cryptographic entropy through periodic key exchanges. Megolm deliberately sacrifices backward secrecy to achieve efficient group communication scaling, implementing only forward secrecy through configurable session rotation policies. These architectural differences create distinct computational complexities and forensic implications, with Olm requiring $O(n^2)$ operations for n participants while Megolm achieves $O(n)$ scaling, fundamentally affecting the volume and distribution of recoverable digital evidence as detailed in Table 1.

5. Methodology

5.1. Experimental environment setup

The experimental environment was designed to replicate typical Matrix homeserver deployment scenarios. A virtual server instance with 2 vCPU cores and 4 GB RAM running Ubuntu 22.04 LTS was configured for this research. This hardware specification reflects common deployment patterns where individuals utilize modest virtualized environments rather than dedicated high-performance servers. The selected specifications represent a realistic baseline for personal or small-group Matrix deployments, enabling assessment of forensic artifacts under typical operational conditions.

5.2. Server synapse configuration

The Matrix homeserver was deployed using Synapse version 1.133.0 through the official Docker Compose configuration with default settings (“matrixdotorg/synapse).

Synapse Admin Panel version 0.11.1 was installed for server management. The homeserver maintained default security settings and used SQLite database storage (“GitHub).

5.3. Element client configuration

Two client platforms were configured to generate diverse forensic artifacts across different operating systems and implementation approaches. The experimental setup utilized Element desktop client running on Ubuntu 22.04 LTS for the PC environment, mobile client version 1.6.44 on Android 13 (“Release Element Android v1). This cross-platform configuration enabled comprehensive evaluation of communication patterns and forensic evidence generation across different client implementations.

5.4. Experimental test cases

We designed multiple test scenarios to generate realistic forensic artifacts using three user accounts to simulate real-world communication patterns. We conducted various interactions between the users to produce diverse forensic evidence, as detailed in Table 2.

Table 3
Authentication and session management artifacts.

Log Category	Available Artifacts	Example Log Content
User Login	User ID Device ID	[timestamp] - synapse.handlers.auth - INFO - Logging in user @user:matrix.example.com on device [Device ID]
Device Registration	IP Address User Agent Platform Details	[timestamp] - synapse.access.http - INFO - [IP] - [PORT] - Processed request: [duration] “POST/_matrix/client/v3/login" “Mozilla/5.0 … Element/1.11.111″

Table 4
Message transmission metadata artifacts.

Log Category	Available Artifacts	Example Log Content
Encrypted Message Send	Room ID Event ID Send User Timestamp	[timestamp] - synapse.access.http - INFO - [IP] - {@user} Processed request: … “PUT/_matrix/client/v3/rooms/[room_id]/send/m.room.encrypted/[event_id]"
Message Receipt Confirm	Read User Message Event ID Read Timestamp	[timestamp] - synapse.access.http - INFO - [IP] - {@user} Processed request: … “POST/_matrix/client/v3/rooms/[room_id]/receipt/m.read/[event_id]"
Typing Indicators	Typing User Room ID Typing State	[timestamp] - synapse.access.http - INFO - [IP] - {@user} Processed request: … “PUT/_matrix/client/r0/rooms/[room_id]/typing/[@user]"

6. Server-side artifacts

6.1. Synapse log analysis

The log examination focused on log files located within the Docker container filesystem at:/var/lib/docker/containers/[container_id]/[container_id]-json.log, where container identifiers correspond to the active Synapse instances.

Our analysis reveals distinct patterns of user activity, device management, and communication metadata that persist across encrypted sessions.

6.1.1. User authentication and session management

Matrix server logs document user authentication activities through comprehensive session management records. As shown in Table 3, login events capture user identifiers, device identifiers, and timestamps for each authentication attempt, while device registration logs preserve network information including IP addresses, port numbers, and client user-agent strings that identify specific application versions.

6.1.2. Message activity

Although Matrix's end-to-end encryption prevents access to message content, server logs retain detailed metadata about communication activities. Table 4 demonstrates that encrypted message transmission logs capture room identifiers, unique event identifiers, sender information, timestamps for each message, message receipt and typing indicator logs reveal real-time user interaction patterns within conversations.

6.1.3. Media file transfer operations

The server logs also contain records of all file transfer activities within the Matrix media repository system. Table 5 demonstrates that file upload operations record storage paths on the server filesystem,

Table 5
Media transfer operation artifacts.

Log Category	Available Artifacts	Example Log Content
File Upload	Storage Path Media URI File Size	[timestamp] - synapse.media.media_repository - INFO - Stored local media in file '/data/media_store/local_content/[path]'
	Upload User Timestamp	[timestamp] - synapse.rest.media.upload_resource - INFO - Uploaded content with URI 'mxc://[server]/[media_id]'
File Download	Download User File Size Transfer Duration Media URI	[timestamp] - synapse.access.http - INFO - [IP] - {@user} Processed request: [duration] … "GET/_matrix/client/v1/media/download/[server]/[media_id]"

Table 6
VoIP service Configuration artifacts.

Log Category	Available Artifacts	Example Log Content
TURN Server Query	Requesting User Query Time Service Endpoint	[timestamp] - synapse.access.http - INFO - [IP] - {@user} Processed request: [duration] … "GET/_matrix/client/v3/voip/turnServer"
VoIP Capability Check	User ID Client Version Protocol Support	[timestamp] - synapse.access.http - INFO - [IP] - {@user} Processed request: [duration] … "GET/_matrix/client/r0/voip/turnServer"

Table 7
Cryptographic operation artifacts.

Log Category	Available Artifacts	Example Log Content
One-Time Key Upload	Device ID Key Type Key Count Timestamp	[timestamp] - synapse.handlers.e2e_keys - INFO - Adding one_time_keys dict_keys (['signed_curve25519:AAA … ']) for device 'BXQYWKGTNU' for user '@user:server'
Device Key Update	User ID Device ID Update Timestamp	[timestamp] - synapse.handlers.e2e_keys - INFO - Updating device keys for device '[device_id]' for user @user:server at [epoch_time]
Cross-Signing Setup	Master Key ID Key Type Configuration Time	[timestamp] - synapse.access.http - INFO - [IP] - {@user} Processed request: [duration] … "PUT/_matrix/client/v3/user/[@user]/account_data/m.secret_storage.key.[key_id]"
Key Backup Operations	Backup Version Room Keys Operation Status	[timestamp] - synapse.access.http - INFO - [IP] - {@user} Processed request: [duration] … "PUT/_matrix/client/v3/room_keys/keys?version = 1″

unique media URIs, file sizes, uploading user identifiers, and timestamps. Download operations similarly log the downloading user, file size, transfer duration, and the accessed media URI.

6.1.4. Voice and video communication preparation

While actual voice and video content transmits through peer-to-peer WebRTC connections outside server visibility, preparatory operations generate identifiable log patterns. Table 6 shows that these preparatory operations log the requesting user identifier, query timestamps, and service endpoint requests.

6.1.5. Cryptographic key management

The cryptographic key management system generates detailed server logs that document encryption infrastructure activities. As illustrated in Table 7, these log records one-time key uploads including device identifiers and key types, device key updates with corresponding timestamps, cross-signing configurations that establish trust relationships between devices, and key backup operations that track encryption key storage.

Table 8
User identification artifacts in synapse database.

Artifact Type	Column Name	Source Table	Example Data
User Identifier	user id	user ips devices	@test:matrix.test.com
Device Identifier	device id	user ips devices	UUAUVUREUO
Device Name	display name	devices	Element Desktop: Linux
IP Address	ip	user ips devices	183.194.74.100
User Agent	user agent	user ips devices	Mozilla/5.0 (X11; Linux x86_64) …

Table 9
Private room relationships.

Artifact Type	Column Name	Source Table	Example Data
User Pair	user id other user id	users who share private rooms	@test:matrix.test.com @userA:matrix.test.com
Room Identifier	room id	users who share private rooms	!KcxiiMqUHMdWKIXuTP:matrix.test.com
Display Names	Display name	room memberships	test
Membership Status	membership	room memberships	join invite

6.2. Synapse database analysis

The Synapse Homeserver stores its default database in SQLite format at*/data/homeserver.db* within the Docker container environment. This database file contains 175 tables including user authentication records, message metadata, room memberships, media references, cryptographic key exchanges, and activity logs. Direct access to this database file can be obtained through Docker commands (*docker cp synapse:/data/homeserver.db ./*) or by mounting the data volume during forensic acquisition. The following sections provide detailed analysis of the database schema structures and their forensic significance.

6.2.1. User identification information

The *user ips* and *devices* tables contain comprehensive identification artifacts as presented in Table 8, including user & device identifiers, device names, IP addresses, and user-agents that collectively enable investigators to establish definitive connections between digital identities and physical individuals.

6.2.2. User relationship

The relationship between users within Matrix server can be examined through room membership data and communication patterns. The

Table 10
Group room relationships.

Artifact Type	Column Name	Source Table	Example Data
Room Statistics	joined members	room stats current	3
	invited members		0
	left members		0
Room Encryption	is encrypted	sliding sync joined rooms	1
Room Name	room name	sliding sync joined rooms	matrix experiment
Membership Events	membership	sliding sync membership snapshots	$qn_NtI-5iBaBRR1QOQS7Eb … MH7gOwFObA

room memberships, *local current membership*, *users who share private rooms*, and *room stats current* tables provide information for mapping both private conversations and group communications.

In Tables 9 and 10, the bidirectional entries in *users_who_share_private_rooms* indicate reciprocal relationship mapping, while *joined_members* count in *room_stats_current* distinguishes private conversations (typically 2 members) from group communications (3 or more members).

The *room_memberships* table serves as the primary forensic artifact for relationship analysis, containing complete records of all user participations with temporal sequencing through *event_stream_ordering* that enables reconstruction of chronological relationship evolution and group formations.

6.2.3. Message content

Rooms configured without E2EE store plaintext message content in the *event_search_content* table, where *c3key* designates the content field type and *c4value* contains actual message text. As shown in Tables 11 and 12, message artifacts include event identifiers for tracking individual messages, type classifications distinguishing room operations (m.room.create) from user content (m.room.message), and device metadata in *internal_metadata* fields that establish message provenance.

The *event_relations* table preserves message linkages through *relates_to_id* fields, enabling reconstruction of threaded conversations. User interactions manifest as reaction content in *aggregation_key* fields and read receipts with millisecond-resolution timestamps.

7. The SynExtract tool

To support rapid server-side forensic artifact extraction from suspect Synapse servers, we developed SynExtract, an open-source Python-based forensic analysis tool available at Github (redundan3y SynExtract). SynExtract provides automated extraction and analysis capabilities through a modular architecture consisting of four primary components: database forensics module (db_forensics) for database artifact extraction, log forensics module (log_forensics) for Docker container log parsing, report generation module (report) supporting

Table 13
Database forensic report categories.

Worksheet	Primary Fields	Forensic Value
User Info	User ID, Creation Time, is Admin, Deactivated	event search content
User IP Records	User ID, Device_ID, IP, User Agent, Last Seen	User, Network and Device information correlation
Device Info	User ID, Device Id, Device name, IP, User Agent, Last Seen	Multi-device usage patterns
Room Info	Room ID, Room Name, Creator, Is Encrypted, Member Counts	Communication channel structure
Room Members	Room ID, User ID, Inviter, Membership, Display Name, Timestamp	Participation timeline and relationships
Chat Messages	Event ID, Room ID, Sender, Message Type, Message Content, Timestamp	Plain Text Chat Message
User Activity Stats	User ID, Event Count, Message Count, Active Rooms, First Activity, Last Activity	Behavioral pattern analysis
.	Media ID, File type, File Size, Uploader, Created Time, Last Access	File transfer documentation

Table 11
Group room membership analysis.

Artifact Type	Column Name	Source Table	Example Data
Event Identifier	event id	events event json event relations	$5qr8cYtaSoQEzmuZDvCM9jX7U6vnPQ4d50ANm2D-vTk
Message Type	type	events	m.room.create m.room.message
Room Context	room id	events event json	!cPjsVTKCjDgQlGNQgF:matrix.test.com
Device Metadata	internal metadata	event json	{""device_id"":""JOGKWUVOGV" … "m.room.create"",""sender"":""@test:matrix.test.com"",""content"":{""room_version"":""10"" ….

Table 12
Message relationship and interaction artifacts.

Artifact Type	Column Name	Source Table	Example Data
Message Relations	relates to id relation type	event relations	$1RyBkfvmqrbeJHjihBlCZ4p8tyIg9Hi2fEm3Iq65TYY m.reference
Reaction Content	aggregation key	event relations	□
Read Receipts	receipt type user id event id	receipts linearized	matrix.test.com,m.read @test:matrix.test.com $1kw_ … –09TdFO2S1lZNk
Receipt Timestamp	data	receipts linearized receipts graph	{"ts":1758031846380}
Plaintext Content	c3key c4value	event search content	content.body plain text message

Table 14
User IP records report.

User ID	Device ID	IP Address	User Agent	Last Seen
@wushufan: matrix.test.com	KLCVGFXBUJ	183.193.178.254	Element/ 1.6.44 (HUAWEI ALN-AL00 …	2025-09-16 22:09:36

Table 15
Room members report.

Room ID	User ID	Inviter	Membership	Display Name
!FcEfqvZdUSTHqrlbsI: matrix.test.com	@wushufan: matrix.test.com	@test: matrix.test.com	invite	wu

Table 16
Log forensic report categories.

Worksheet	Primary Fields	Forensic Value
IP Access Statistics	IP, Count, First Seen, Last Seen, User Count, Device Count, Users, User Agent, Top Paths.	Network activity profiling
Users (from logs)	User ID, First Seen, Last Seen, IP Count, IP Addresses, Device Count, Room Accessed, API Calls, User Agents	Temporal activity reconstruction
Rooms (from logs)	Room ID, User Count, Users, First Activity, Last Activity, Message Count, Event Count	Communication channel activity
Matrix API Calls	Time, IP, User Id, Device ID, Room ID, API Type, method, Path, Status	API interaction timeline
Authentication Logs	Time, Type, IP Address, User Agent, Status, Success	Authentication related access log
Access Logs	IP Address Time, Method, Path, Status, User Agent	Full access log

both Excel and HTML output formats, and an interactive user interface module (UI) with bilingual support.

The database forensics module read Synapse's SQLite database and extract artifacts across ten categories including user accounts, access records, devices, rooms, memberships, messages, and media transfers as shown in Table 13.

Table 14 shows an example User IP Records worksheet extracted by SynExtract, demonstrating the correlation between user identities, network addresses, device identifiers, and User-Agent strings.

Table 15 presents the Room Members worksheet structure, revealing temporal evolution of room participation through membership status transitions and invitation chains.

The log forensics module processes Docker logs, parsing structured entries to extract authentication events, API access patterns, and error conditions. This module reconstructs user identities, IP addresses, room participation, and temporal activity patterns entirely from log data without requiring database access. Table 16 details the log extraction categories, demonstrating how investigators can recover essential forensic artifacts including user profiles, device information, and room activities from server logs alone.

SynExtract generates three interactive HTML visualizations: room relationship graphs displaying bipartite networks between rooms and users, user interaction graphs showing connections weighted by shared room participation, and room activity timelines presenting temporal event distributions.

Fig. 3 shows the room relationship visualization with rooms as red square nodes and users as blue circular nodes, enabling identification of communication hubs and clustering patterns.

Fig. 4 presents the room activity timeline extracted from logs, displaying chronological user interactions color-coded by participant to reveal coordination patterns.

8. Discussion

This research presents the first comprehensive forensic examination of Matrix's server-side artifacts, specifically focusing on the Synapse Homeserver implementation across 175 database tables and structured log entries. We developed the SynExtract forensic tool for investigators to systematically recover and analyze these artifacts.

However, this research acknowledges critical limitations requiring future investigation. Our experimental scope was deliberately constrained to single-server deployments to establish foundational forensic methodologies. Consequently, forensic artifacts specific to federated communications and bridged external platforms were not examined.

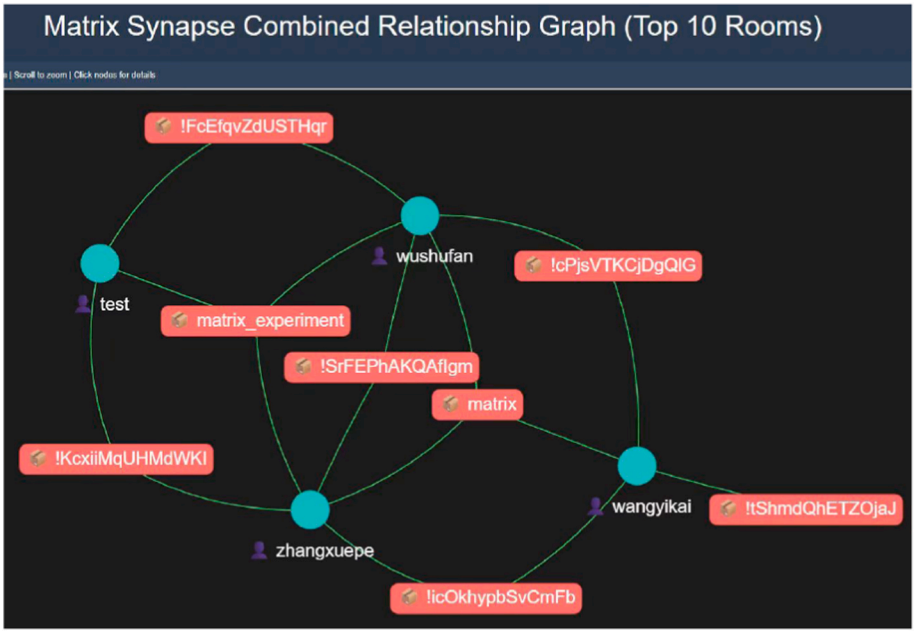

Fig. 3. Room relationship graph.

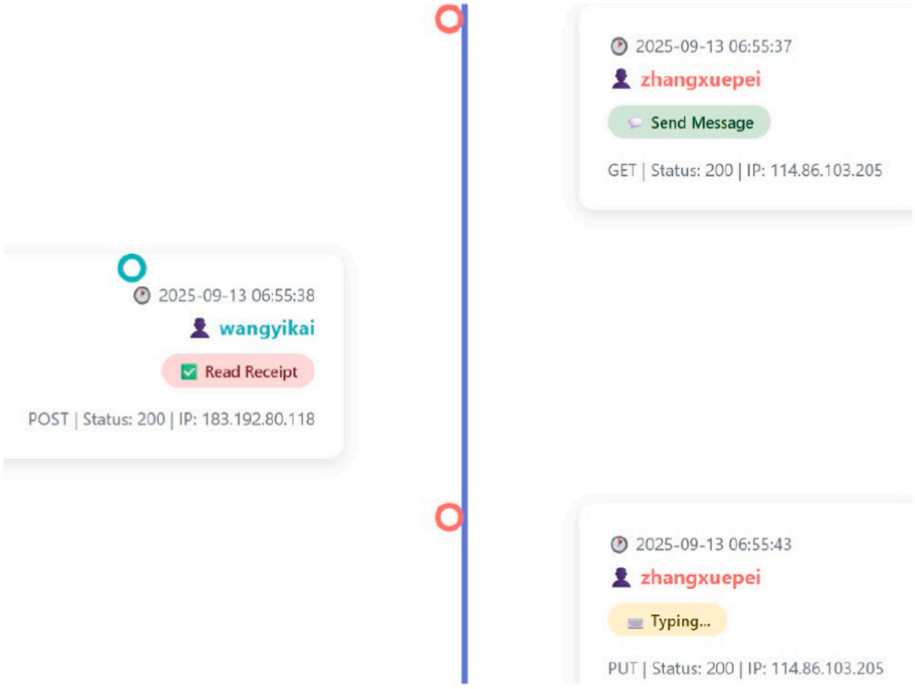

Fig. 4. Room timeline report.

Our analysis also focused exclusively on passive artifact recovery without exploring administrative interception of cryptographic key material during active sessions. Additionally, comprehensive forensic analysis of Element clients across desktop, mobile, and web platforms remains unexplored.

Future research should address these limitations by: (1) examining the feasibility of message interception during active sessions with administrative privileges, and (2) conducting systematic forensic investigations across all Element client platforms including desktop, mobile, and web implementations.

Acknowledgements

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Data Privacy & Encryption Statistics 2026. Global trends & forecast [Online]. Available: https://comparecheapssl.com/data-privacy-encryption-statistics/. (Accessed 16 December 2025).

M. R. Albrecht, S. Celi, B. Dowling, and D. Jones, "Practically-Exploitable Cryptographic Vulnerabilities in Matrix.".

Albrecht, M.R., Dowling, B., Jones, D., 2024. Device-oriented group messaging: a formal cryptographic analysis of matrix' core. In: Proc IEEE Symp Secur Priv, pp. 2666–2685. https://doi.org/10.1109/SP54263.2024.00075.

Christy, B., 2022. Still waiting to Go dark nearly 30 years on: the privacy implications of end-to-end encryption (E2EE). Public Interest Law Journal of New Zealand 9 [Online]. Available: https://heinonline.org/HOL/Page?handle=hein.journals/piljnz9&id=29&div=&collection=. (Accessed 12 September 2025).

Element. In an increasingly volatile world, governments turn to element and matrix [Online]. Available: https://element.io/blog/in-an-increasingly-volatile-world-governments-turn-to-element-and-matrix/. (Accessed 27 May 2025).

Hartel, P., van Wegberg, R., 2023. Going dark? Analysing the impact of end-to-end encryption on the outcome of Dutch criminal court cases. Crime Sci 12 (1). https://doi.org/10.1186/s40163-023-00185-4.

Jaeckel, L., Spranger, M., Labudde, D., 2025. Forensic analysis of telegram messenger on iOS smartphones. Forensic Sci. Int.: Digit. Invest. 52, 301866. https://doi.org/10.1016/J.FSIDI.2025.301866.

Li, H., Wu, Y., Huang, R., Mi, X., Hu, C., Guo, S., 2023. Demystifying decentralized matrix communication network: ecosystem and security. In: Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS, pp. 260–267. https://doi.org/10.1109/ICPADS60453.2023.00047.

Onik, A.R., Brown, J., Walker, C., Baggili, I., 2025. A systematic literature review of secure instant messaging applications from a digital forensics perspective. ACM Comput. Surv. 57 (9), 1–36. https://doi.org/10.1145/3727641.

Pisaric, M., 2022. Communications encryption as an investigative obstacle' (2022) 60(1) journal of criminology and criminal law (JCCL) 61 MLA 9th ed. Pisaric, Milana. Journal of Criminology and Criminal Law (JCCL) 60 (1), 61–74. https://doi.org/10.47152/rkkp.

Schipper, G.C., Seelt, R., Le-Khac, N.A., 2021. Forensic analysis of matrix protocol and riot.im application. Forensic Sci. Int.: Digit. Invest. 36. https://doi.org/10.1016/j.fsidi.2021.301118.

H. Stenhav, "Security Evaluation of the Matrix Server-Server API.".

Defence Military C3 Interoperable communications [Online]. Available: https://element.io/sectors/defence. (Accessed 26 September 2025).

End-to-end encryption Matrix client tutorial [Online]. Available: https://uhoreg.gitlab.io/matrix-tutorial/encryption.html#olm-and-megolm. (Accessed 12 September 2025).

GitHub - Awesome-Technologies/synapse-admin: admin console for synapse matrix homeserver [Online]. Available: https://github.com/Awesome-Technologies/synapse-admin. (Accessed 19 September 2025).

Matrixdotorg synapse - docker image | docker hub [Online]. Available: https://hub.docker.com/r/matrixdotorg/synapse. (Accessed 19 September 2025).

redundan3y SynExtract · GitHub [Online]. Available: https://github.com/redundan3y/SynExtract. (Accessed 16 December 2025).

Release element android v1.6.44 · element-hq/element-android · GitHub [Online]. Available: https://github.com/element-hq/element-android/releases/tag/v1.6.44. (Accessed 26 September 2025).