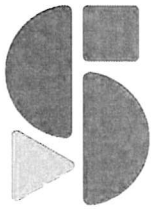
Telegram's MTProto: Assessing Deanonymization Potential for a Network Attacker

GNMX-01

Prepared for Global Network Solutions, Inc.

Dr. Nadim Kobeissi
Symbolic Software



October 17, 2025

Abstract

This report presents a technical assessment of Telegram's MTProto protocol, focusing on the privacy implications of its transport layer design. We examine the persistent device identifier (`auth_key_id`) that appears in every MTProto message header and investigate whether this identifier is exposed to passive network observers through Telegram's transport protocol choices.

Our analysis reveals that both Telegram for Android and Telegram Desktop transmit MTProto over unencrypted TCP connections, exposing the `auth_key_id` to passive observation despite the availability of encrypted transport alternatives. This 64-bit identifier remains constant across application restarts, network changes, and extended time periods, enabling device tracking by any network intermediary positioned between client and server.

We evaluate claims from public reporting about these vulnerabilities, finding the core technical assertions regarding `auth_key_id` exposure and tracking capability to be accurate and reproducible. The exposure affects all Telegram users including those using end-to-end encrypted Secret Chats, as the vulnerability occurs at the transport layer beneath the application-layer encryption.

The implications extend beyond theoretical concerns. The exposure of persistent device identifiers through unencrypted transports enables tracking that persists across network changes, undermines anonymity tools, and creates surveillance opportunities for a broad range of adversaries including ISPs, network administrators, and state-level actors. This is particularly significant given Telegram's adoption by journalists, activists, and other high-risk users.

We conclude that this vulnerability results from Telegram's architectural decision not to mandate transport-layer encryption, despite this being standard practice among competing platforms. The technical solution—implementing mandatory TLS for all MTProto connections—would eliminate the tracking capability while requiring minimal implementation changes. Until such measures are implemented, Telegram bears responsibility for the privacy implications of exposing persistent device identifiers to network observers.

Contents

1 | Executive Summary 1

2 | Introduction 5

2.1 | Claims Under Investigation 6

2.2 | Scope and Methodology 9

2.3 | Report Structure 11

2.4 | Disclaimer 12

3 | Target Assessment 13

3.1 | Telegram's MTProto 14

3.1.1 MTProto's Transports 15

3.1.2 MTProto's `auth_key_id` 15

3.1.2.1 Privacy Questions Raised by `auth_key_id` 16

3.1.2.2 Perfect Forward Secrecy and `auth_key_id` 17

3.2 | Assessing `auth_key_id` Exposure 20

3.2.1 Telegram for Mobile (Android) 20

3.2.2 Telegram for Desktop (macOS) 21

3.3 | Findings 23

3.4 | Implications of Findings 25

4 | Critical Evaluation 29

4.1 | Claim 1: `auth_key_id` Enables User Tracking 30

4.1.1 Technical Premises 30

4.1.2 Evaluation Against Evidence 30

4.1.3 Assessment 31

4.2 | Claim 2: Metadata Surveillance 33

4.2.1 Technical Premises 33

4.2.2 Evaluation Against Evidence 33

4.2.3 Assessment 35

4.3 | Claim 3: E2EE Does Not Prevent Tracking 37
4.3.1 Technical Premises 37
4.3.2 Evaluation Against Evidence 37
4.3.3 Assessment 38

4.4 | Claim 4: Infrastructure & Surveillance 40
4.4.1 Technical Premises 40
4.4.2 Evaluation Against Evidence 40
4.4.3 Assessment 42

4.5 | Claim 5: Router Control Enables Tracking 46
4.5.1 Technical Premises 46
4.5.2 Evaluation Against Evidence 46
4.5.3 Assessment 47

4.6 | Summary of Critical Evaluation 48

5 | Privacy Questions & Answers 50

5.1 | Infrastructure Layer Separation 52

5.2 | Feasability of Global Correlation 55
5.2.1 Scale and Complexity Challenges 55
5.2.2 Operational and Coordination Barriers 56
5.2.3 Realistic Threat Scenarios 57
5.2.4 Assessment and Implications 57

5.3 | Varied Surveillance Environments 59
5.3.1 High-Surveillance Environments: Beyond IP Attribution 59
5.3.2 Privacy-Protective Environments: Enabling Targeted Surveillance ... 61
5.3.3 Comparative Assessment and Implications 63

5.4 | Illustrative Case Study 66
5.4.1 Hypothetical Scenario: Journalist Deanonymization Timeline 66
5.4.2 Key Implications of This Scenario 68

6 | Architectural Responsibility and Liability 70

6.1 | The Root Cause: Telegram's Design Choice 70

6.2 | Misplaced Liability 71

6.3 | The Liability Transfer 72

6.4 | False Equivalence in Attribution 73

6.5 | Industry Standards and Best Practices 74

6.6 | The Simple Solution 75

7 | Conclusions 76

7.1 | Evaluation of Public Claims 78

7.2 | Privacy Impact Analysis 79

7.3 | Architectural Responsibility 80

7.4 | Implications and Recommendations 81

7.5 | Limitations of This Study 83

7.6 | Conclusion 84

8 | About Symbolic Software 85

Bibliography 86

Appendix A | Legal Context 88

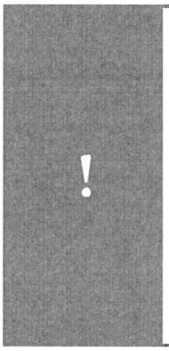
List of Acronyms

API	Application Programming Interface	14
GDPR	General Data Protection Regulation	28
HTTP	Hypertext Transfer Protocol	14
HTTPS	Hypertext Transfer Protocol Secure	13
ISP	Internet Service Provider	1
OCCRP	Organized Crime and Corruption Reporting Project	1
PFS	Perfect Forward Secrecy	1
TCP	Transmission Control Protocol	1
TLS	Transport Layer Security	3
UDP	User Datagram Protocol	14

1

Executive Summary

For the busy executive who just doesn't have the time!



Legal & Ethical Disclaimer

This report describes technical feasibility only. Any references to "network observers", "visibility to networks", or "logging" refer to technical potential under specific conditions and do not imply lawful or routine practice. The interception, capture, or retention of communications or transport-level metadata (including headers such as `auth_key_id`) without proper legal authorisation — for example a court order or other statutory basis — is unlawful in many jurisdictions. This report does not encourage or endorse any unlawful interception or monitoring; it is intended purely to assess architectural privacy risks and to recommend mitigations.

This report presents an independent technical analysis of claims made in a June 2025 investigation by the Organized Crime and Corruption Reporting Project (OCCRP) regarding potential security vulnerabilities in Telegram's messaging platform. The investigation alleged that Telegram's MTProto protocol exposes a persistent device identifier (the *authorization key identifier*, or `auth_key_id`) through unencrypted network connections, enabling tracking of users by passive network observers.

Our analysis confirms the core technical findings: Telegram clients on both Android and macOS/Desktop platforms transmit MTProto messages over unencrypted Transmission Control Protocol (TCP) connections, exposing the `auth_key_id` in cleartext (or trivially obfuscated form) to any network intermediary. The `auth_key_id` functions as a persistent device identifier that remains constant across sessions, IP address changes, network switches, and geographic locations. This creates a technical capability for device tracking by any entity with passive network access, including Internet Service Provider (ISP)s, network administrators, government surveillance programs, and other adversaries positioned along the communication path.

We validated that this tracking capability persists even when users enable Telegram's Perfect Forward Secrecy (PFS) feature or use Secret Chats with end-to-end encryption, because the `auth_key_id` exposure occurs at the MTProto transport layer beneath these security features. Temporary authorization keys introduced by PFS are

equally observable and can be linked across key rotations through timing correlation, providing no meaningful protection against network-level tracking.

Our empirical testing demonstrated that `auth_key_id` rotation, while theoretically possible, occurs infrequently in practice—we observed no rotation across application restarts, network changes, or extended time periods in our tests. This persistence contradicts claims of frequent rotation and significantly amplifies the privacy implications of the exposure.

Privacy Impact Assessment

The privacy implications of `auth_key_id` exposure vary significantly across different threat models and surveillance environments, as detailed in our Privacy Questions & Answers analysis (Chapter 5):

Infrastructure Layer Considerations: While pure Layer 1 infrastructure providers may not have immediate access to Layer 3 traffic containing `auth_key_id` values, the technical barriers to such access are surmountable with moderate investment in monitoring equipment. More critically, focusing solely on specific infrastructure layers understates the exposure risk, as traffic is readily observable at numerous other points in the network path where Layer 3 access is routine.

Global Correlation Feasibility: While comprehensive, real-time correlation of all one billion Telegram users' `auth_key_id` values is technically and operationally implausible given the scale, network diversity, and coordination required, more realistic threat scenarios remain concerning. These include targeted surveillance of specific individuals, regional or national-scale tracking within centralized infrastructure, retrospective analysis using retained logs, and opportunistic collection by various actors.

Surveillance Environment Variations: The intelligence value of `auth_key_id` differs but remains significant across all surveillance environments:

- In high-surveillance jurisdictions with pervasive IP attribution, `auth_key_id` provides device-level granularity, cross-network tracking continuity, retroactive analysis capabilities, and circumvention detection that IP addresses alone cannot achieve.
- In privacy-protective jurisdictions, `auth_key_id` enables targeted tracking following initial identification, opportunistic correlation through various touchpoints, behavioral fingerprinting, and gradual intelligence accumulation that would otherwise be legally restricted.

Our illustrative case study (Chapter 5) demonstrates how a single identity anchor point—such as a journalist registering for conference WiFi—can compromise all past

and future communications using that `auth_key_id`, enabling comprehensive retroactive surveillance using only standard logging infrastructure already deployed by service providers.

Architectural Responsibility

Chapter 6 establishes that responsibility for this vulnerability lies squarely with Telegram, not with infrastructure providers who handle its traffic. By choosing not to implement industry-standard Transport Layer Security (TLS) or similar transport encryption, Telegram:

- Exposes sensitive metadata to every network intermediary along the communication path
- Transfers liability for user privacy to infrastructure partners who have no control over protocol design
- Forces datacenter operators, ISPs, and transit providers to handle unencrypted traffic containing trackable identifiers
- Creates legal, reputational, and operational burdens for partners who never requested visibility into user metadata
- Violates established security principles of defense in depth, least privilege, and cryptographic protection

The investigation's focus on infrastructure provider relationships, while having investigative merit, creates a false equivalence that obscures the fundamental issue: the vulnerability exists because of Telegram's protocol design, not because of who operates the infrastructure. If Telegram implemented proper transport-layer encryption, the identity and trustworthiness of infrastructure providers would be largely irrelevant to user privacy at the network layer.

Key Findings and Recommendations

The investigation's technical claims regarding `auth_key_id` visibility, persistence, and tracking capability are substantiated by our independent analysis. However, the investigation's attempts to attribute this vulnerability to specific adversaries rely on circumstantial evidence involving business relationships rather than direct proof of surveillance capability or intent.

Organizations and individuals making security decisions about Telegram should understand:

- Device tracking through `auth_key_id` observation is technically feasible for any passive network observer
- This capability cannot be mitigated through user configuration changes or security features
- The vulnerability affects all Telegram users regardless of their use of Secret Chats or PFS
- Standard privacy tools like VPNs provide no protection against `auth_key_id` exposure
- The persistence of these identifiers enables both real-time tracking and retroactive surveillance

The proper solution is not to debate which infrastructure providers might exploit this visibility or to assess the likelihood of surveillance at different network layers. The solution is for Telegram to eliminate the vulnerability entirely through mandatory use of transport-layer encryption—a standard practice adopted by virtually every other major messaging platform. The technical implementation is straightforward, the performance impact negligible, and the privacy benefits substantial.

Until Telegram implements proper TLS encryption, it continues to expose its users to unnecessary privacy risks and saddles its entire ecosystem of infrastructure partners with liability for metadata that should never be their responsibility. This represents not merely a technical failing, but a fundamental abdication of Telegram's responsibility to protect user privacy through appropriate cryptographic measures.

2

Introduction

The proliferation of secure messaging applications has fundamentally transformed global communications, promising users unprecedented levels of privacy and security in an increasingly surveilled digital landscape. Among these platforms, Telegram has emerged as a dominant force, boasting over one billion monthly active users and a reputation built on its founder Pavel Durov's alleged defiance to government authority and commitment to user privacy. Yet recent investigative reporting has raised questions about the actual security guarantees provided by Telegram's technical infrastructure, particularly regarding the potential for user tracking and deanonymization through seemingly innocuous protocol elements.

On June 10, 2025, the OCCRP published a comprehensive investigation titled *"Telegram, the FSB, and the Man in the Middle,"* [1] which presented alarming claims about vulnerabilities in Telegram's security architecture and potential connections between its infrastructure providers and Russian intelligence services. The investigation's findings, if accurate, would represent a fundamental breach of trust for millions of users who rely on Telegram for secure communications, including journalists, activists, opposition figures, and ordinary citizens seeking privacy from authoritarian surveillance.

This technical report represents an independent, rigorous analysis commissioned to examine the veracity and technical feasibility of the most critical claims presented in the OCCRP investigation. **Our analysis focuses specifically on the `auth_key_id` component of Telegram's MTProto protocol: the element identified by the OCCRP investigation as a potential vector for user tracking and deanonymization.** Through comprehensive protocol analysis, network traffic examination, and threat modeling, we seek to establish clear technical distinctions between theoretical vulnerabilities and practical exploitation capabilities.

2.1 | Claims Under Investigation

The OCCRP investigation presents several interconnected claims that warrant careful technical scrutiny. This report will present a full technical investigation of these claims with the aim of evaluating their truthfulness through a careful analysis of the evidence.

Claim 1: `auth_key_id` Enables User Tracking
The article states:

> *"The unencrypted part is called 'auth_key_id.' This makes it possible to identify a specific user device. [...] If I know your device's 'auth_key_id,' and I can listen in on the network that handles the data... I know it is your specific device communicating with Telegram servers."*
>
> — **OCCRP** quoting internal security expert Michał "rysiek" Woźniak

This claim suggests that the `auth_key_id`, transmitted in plaintext as part of the MTProto protocol, serves as a persistent device identifier that could be used to track users across sessions and locations.

Claim 2: Metadata Collection
According to the investigation:

> *"By looking at the network packets... I also get your IP address at a given time, which tells me your rough geographic location. [...] In such an attack, the hackers aren't even interested so much in the user's correspondence. They get metadata to analyze. And that means IP addresses, user locations, who exchanges data packets with whom, the kind of data it is... really, all possible information."*
>
> — **OCCRP** quoting internal security expert Michał "rysiek" Woźniak

Claim 3: E2EE Does Not Prevent Tracking
The investigation makes the striking assertion that:

> *Even users who use its 'end-to-end' encryption features are vulnerable to being tracked by anyone who can monitor its network traffic.*
>
> — OCCRP

This claim challenges the fundamental security assumptions of Telegram's Secret Chat feature, which purports to provide complete privacy through end-to-end encryption.

Claim 4: Infrastructure Control Enables Surveillance
The article further claims the following:

> *If someone has access to Telegram traffic and cooperates with Russian intelligence services, this means that the device identifier becomes a really big problem — a tool for global surveillance of messenger users, regardless of where they are and what server they connect to.*
>
> — OCCRP quoting internal security expert Michał "rysiek" Woźniak

This claim suggests that control over Telegram's network infrastructure could enable comprehensive, global tracking of users.

Claim 5: Router Control Enables Tracking
The investigation states:

> *If a company controls the routers that distribute traffic passing through Telegram servers, this means that it, or anyone to whom it grants such access, can see the identifiers of messenger users.*
>
> — **OCCRP** quoting internal security expert Michał "rysiek" Woźniak

This claim implies that network-level control at specific infrastructure points could compromise user privacy even without breaking encryption.

2.2 | Scope and Methodology

This report focuses on a specific, well-defined technical question: whether Telegram's MTProto protocol exposes persistent device identifiers (`auth_key_id` values) to passive network observers, and whether such exposure enables device tracking across sessions, network changes, and geographic locations.

Our methodology combines protocol analysis, empirical network traffic examination, and independent replication of prior security research. Specifically, we:

1. **Protocol documentation review:** We examined Telegram's official MTProto protocol documentation [2–4] to understand the architectural role of `auth_key_id`, the structure of MTProto messages, and the available transport options.
2. **Literature review:** We analyzed independent security research on Telegram's protocol implementation, with particular focus on Michał "rysiek" Woźniak's technical analysis [5] of Telegram for Android.
3. **Empirical traffic analysis:** We conducted original packet capture and analysis of Telegram Desktop on macOS, examining network traffic patterns during various usage scenarios to determine which transport protocols are employed in practice and whether `auth_key_id` values are observable.
4. **Independent replication:** We replicated key findings from prior research to validate their accuracy and ensure our conclusions are not dependent on a single source or methodology.
5. **Claim evaluation:** We systematically evaluated specific technical claims from the OCCRP investigation against our empirical findings and established security principles.

What this report addresses:

- The technical architecture of MTProto and the role of `auth_key_id`
- Whether `auth_key_id` values are exposed to passive network observers in practice
- Whether `auth_key_id` values function as persistent device identifiers
- The privacy implications of `auth_key_id` exposure for device tracking
- The accuracy of specific technical claims regarding these issues
- How `auth_key_id` exposure varies across different surveillance environments

- The practical feasibility of global correlation at Telegram's scale
- Real-world scenarios demonstrating exploitation of this vulnerability
- The architectural responsibility for this security weakness

What this report does not address:

- Cryptographic analysis of MTProto's encryption algorithms
- Security of Telegram's server infrastructure or backend systems
- Analysis of Telegram's corporate structure or business relationships
- Geopolitical assessment of specific adversaries' capabilities or intentions
- Comprehensive security audit of all Telegram features and protocols
- Comparative analysis of Telegram versus competing messaging platforms

Our analysis is deliberately narrow and technical, focusing on empirically verifiable questions about protocol behavior and network observability rather than broader questions of organizational trust, geopolitical context, or comprehensive security posture.

2.3 | Report Structure

This report is organized into five principal chapters:

Chapter 2 (Introduction) provides context for the investigation, presents the specific claims under examination, and establishes the scope and methodology of our analysis.

Chapter 3 (Target Assessment) presents our technical analysis of Telegram's MTProto protocol, examining its architecture, transport options, and the nature of the `auth_key_id` component. We review independent research on Telegram for Android and present our original empirical analysis of Telegram Desktop on macOS, documenting our findings regarding transport protocol selection and `auth_key_id` exposure. This chapter establishes the technical facts that form the foundation for subsequent evaluation.

Chapter 4 (Critical Evaluation) evaluates the five principal claims from the OCCRP investigation against the technical evidence established in Chapter 3. For each claim, we assess its technical accuracy, identify any elements of overstatement or speculation, and provide qualified conclusions distinguishing between empirically supported findings and inferential reasoning.

Chapter 5 (Privacy Questions & Answers) addresses critical privacy questions that emerge from our technical findings, providing nuanced analysis of how `auth_key_id` exposure translates into real-world surveillance capabilities. Through examination of different threat models, infrastructure layers, and deployment contexts, we assess the practical implications of this vulnerability while maintaining clear distinction between theoretical possibilities and operational realities. The chapter includes an illustrative case study demonstrating both the serious privacy risks and the step-by-step process through which deanonymization can occur using only standard network infrastructure and commercially available tools.

Chapter 6 (Architectural Responsibility and Liability) examines the fundamental question of responsibility for the `auth_key_id` exposure vulnerability. Rather than focusing on the trustworthiness of specific infrastructure providers as the OCCRP investigation does, we analyze how Telegram's conscious decision not to implement transport-layer encryption creates this vulnerability and inappropriately transfers liability to every network intermediary. This chapter demonstrates that the solution is not more careful selection of infrastructure partners but rather the implementation of industry-standard transport encryption that would render such trust unnecessary.

2.4 | Disclaimer

This independent technical analysis was commissioned under strict conditions guaranteeing complete independence and editorial autonomy for Symbolic Software. The terms of engagement explicitly provide that:

- Symbolic Software retains full editorial control over all findings, conclusions, and recommendations presented in this report;
- Symbolic Software has the unconditional right to reach any conclusion deemed technically appropriate based on the evidence, regardless of whether such conclusions align with the expectations or preferences of any party;
- No party may modify, censor, or otherwise interfere with the technical content or conclusions of this report;
- Symbolic Software maintains sole authority over the methodology, scope, and presentation of the technical analysis;
- The integrity of this report and its findings shall be preserved in any publication or distribution.

These conditions were agreed upon as prerequisites for undertaking this analysis. All findings and conclusions presented herein represent the independent technical judgment of Symbolic Software based solely on empirical evidence, protocol analysis, and established security research principles.

3

Target Assessment

This chapter provides a technical assessment of Telegram's MTProto protocol with particular focus on the privacy implications of its transport layer design choices. We examine the structure and behavior of MTProto's authorization key identifier (`auth_key_id`), a persistent device identifier that appears in the header of every MTProto message, and investigate whether this identifier is exposed to passive network observers through Telegram's choice of transport protocols.

Our analysis proceeds in several stages. First, we provide an overview of MTProto's architecture, explaining its cryptographic layer, transport options, and the role of the `auth_key_id` in message authentication. We then examine the theoretical privacy risks associated with `auth_key_id` exposure, particularly the potential for this persistent identifier to enable long-term device tracking by network intermediaries.

The central question we address is whether Telegram's implementation choices expose the `auth_key_id` to passive network observation in practice. While MTProto's specification documents support for both encrypted (Hypertext Transfer Protocol Secure (HTTPS)) and unencrypted (TCP) transports, the actual transport protocol used by Telegram clients in real-world deployments determines whether the `auth_key_id` remains protected by transport-layer encryption or is visible to any party capable of monitoring network traffic.

We review independent security research on Telegram for Android and present our own empirical analysis of Telegram Desktop on macOS, examining network traffic patterns to determine which transport protocols are actually employed. Our findings reveal that both clients transmit MTProto over unencrypted TCP connections, exposing the `auth_key_id` to passive observation despite the availability of encrypted transport alternatives.

The implications of these findings extend beyond simple technical observations. The exposure of persistent device identifiers through unencrypted transports affects Telegram's privacy properties in fundamental ways, enabling tracking capabilities that persist across network changes, undermine anonymity tools, and create surveillance opportunities for a broad range of potential adversaries. These implications are particularly significant given Telegram's widespread adoption by journalists, activists, and other high-risk users who depend on secure communications in sensitive contexts.

3.1 | Telegram's MTProto

Telegram defines its own wire protocol, "MTProto" [2], which it terms as a *"basic layer of encryption used for Cloud chats (server-client encryption)"*.[1]

MTProto is a protocol designed for access to a server Application Programming Interface (API) from applications running on mobile devices. The protocol is subdivided into three virtually independent components: a high-level component (API query language) that defines how API queries and responses are converted to binary messages; a cryptographic (authorization) layer that defines how messages are encrypted prior to transmission; and a transport component that defines how the client and server transmit messages over existing network protocols (such as Hypertext Transfer Protocol (HTTP), HTTPS, WebSockets, TCP, or User Datagram Protocol (UDP)). Major Telegram clients currently use MTProto 2.0, with the deprecated MTProto v1.0 being phased out.

The cryptographic layer of MTProto works by adding an external header to each encrypted message consisting of a 64-bit key identifier and a 128-bit message key. The user key together with the message key defines an actual 256-bit key for AES-256 encryption. The message key is derived from the SHA256 hash of the message body (including session data, message ID, sequence number, and padding) prepended by 32 bytes from the authorization key. Authorization keys are typically generated when a client application first runs and rarely change thereafter. To protect against post factum decryption in cases where authorization keys might be compromised (such as through device theft), MTProto supports Perfect Forward Secrecy in both cloud chats and secret chats.

As such, MTProto can be understood as a wire transport protocol equivalent to how WhatsApp employs Noise pipes:

> "
>
> *Communication between WhatsApp clients and WhatsApp chat servers is layered within a separate encrypted channel using Noise Pipes with Curve25519, AES-GCM, and SHA256 from the Noise Protocol Framework for long running interactive connections.*
>
> "
>
> — WhatsApp, Inc. [6]

[1] This positions MTProto in contrast to Telegram's "Secret Chats" feature, which aims to provide end-to-end encryption **over** MTProto.

Similarly, MTProto can be understood as a wire transport protocol equivalent to how Signal employs TLS. In addition, Signal has in the past taken advantage of its usage of TLS as a transport layer in order to perform censorship circumvention by using a technique known as "domain fronting". [7]

3.1.1 | MTProto's Transports

MTProto supports multiple transport protocols for delivering encrypted payloads between client and server. These include TCP, WebSocket, WebSocket over HTTPS, HTTP, and HTTPS. [4]

The TCP transport is the most straightforward implementation, where MTProto payloads are transmitted over plain TCP sockets on ports 80, 443, 5222, or others as specified by the server configuration. TCP connections are unencrypted at the transport layer, with encryption provided solely by MTProto's cryptographic layer. Framing is managed by the MTProto transport protocol itself, and all messages require explicit acknowledgment.

WebSocket and WebSocket over HTTPS transports operate similarly to TCP, treating the connection as a duplex byte stream. WebSocket connections can be established over plain HTTP (port 80) or HTTPS (port 443) using specific URI formats. When using WebSockets, transport obfuscation is required, and framing remains managed by MTProto rather than by WebSocket message boundaries.

HTTP and HTTPS transports are also supported, though WebSocket is recommended for browser-based clients due to its full-duplex streaming capabilities. HTTP transport uses traditional HTTP/1.1 with keepalive over port 80, while HTTPS operates over port 443. Unlike other transports, HTTP framing is handled by the HTTP protocol itself rather than MTProto.

In practice, Telegram clients predominantly default to using the TCP transport despite the availability of HTTPS options. As we will demonstrate in Chapter 4, this preference for unencrypted TCP connections has significant implications for traffic analysis and network observability, since while the MTProto payload itself remains encrypted, the transport layer metadata and connection patterns are fully visible to network observers.

3.1.2 | MTProto's `auth_key_id`

The "authorization key identifier" (`auth_key_id`) is a 64-bit value that forms part of the external header added to every MTProto message. [3] It serves to uniquely identify which authorization key was used to encrypt a message, allowing the server to determine both the encryption key and the associated user.

The `auth_key_id` is derived from the authorization key itself: it consists of the 64 lower-order bits of the SHA-1 hash of the authorization key. Notably, MTProto 2.0

continues to use SHA-1 for this purpose despite otherwise migrating to SHA-256, as the auth_key_id must identify authorization keys independently of the protocol version being used.

Authorization keys themselves are 2048-bit keys shared between a client device and the server, created during initial user registration through a Diffie-Hellman key exchange that occurs entirely on the client device. Importantly, the authorization key is never transmitted over the network. Each authorization key is user-specific, though a single user may possess multiple keys corresponding to different devices or "permanent sessions."

In the event that two different authorization keys produce identical 64-bit identifiers (a hash collision), the authorization key must be regenerated to ensure uniqueness. A special case exists where auth_key_id is zero, which indicates that no encryption is being used—this is only permissible for a limited set of message types used during the initial registration process when establishing the authorization key through Diffie-Hellman exchange.

For regular sessions, once an authorization key has been established, the auth_key_id is included in the header of every encrypted message sent between the client and server. This allows the server to immediately identify which authorization key should be used to decrypt the incoming message without requiring any additional negotiation or key lookup protocol.

3.1.2.1 | Privacy Questions Raised by auth_key_id

Since the auth_key_id is transmitted in plaintext as part of the message header, it is natural to ask whether it can be exploited as a unique and persistent identifier that can be used to track a particular device's communication sessions across different network connections and IP addresses.

The potential for auth_key_id to serve as a tracking identifier raises several critical privacy concerns. If the auth_key_id remains constant across sessions and can be observed by network intermediaries, it could effectively function as a persistent pseudonym that allows tracking of a user's activities even when they change IP addresses, switch networks, or attempt to use anonymity tools like VPNs or Tor.

The severity of this concern is directly modulated by MTProto's choice of transport protocol. If MTProto operates over unencrypted TCP connections, the auth_key_id is transmitted in plaintext and is therefore visible to any network observer positioned between the client and server. This includes ISPs, network administrators, government agencies conducting surveillance, or any entity capable of passive traffic monitoring. In such scenarios, the auth_key_id could be collected, correlated across time and network locations, and used to build profiles of user behavior.

By contrast, if MTProto were transmitted over HTTPS, the entire MTProto message—including the auth_key_id header—would be encrypted at the transport

layer by TLS. This would prevent passive network observers from extracting the `auth_key_id`, significantly reducing the potential for this field to be exploited as a tracking mechanism. Only the Telegram servers themselves, or attackers capable of compromising TLS (through certificate attacks, man-in-the-middle attacks, or compelled cooperation), would be able to observe the `auth_key_id`.

This dynamic transforms the choice between TCP and HTTPS transports from a seemingly minor implementation detail into a fundamental privacy decision. The question is not merely one of defense-in-depth or best practices, but rather whether a persistent, device-specific identifier is exposed to the entire network path or protected from passive observation. Given that authorization keys are designed to be long-lived and survive across sessions, IP address changes, and even application restarts, any tracking capability enabled by `auth_key_id` visibility could be correspondingly persistent and comprehensive.

Furthermore, the implications extend beyond simple session linkability. If an adversary can associate a specific `auth_key_id` with a particular individual (perhaps through an initial observation where the user's identity is known), subsequent observations of that same `auth_key_id` anywhere on the network could reveal that individual's ongoing communication patterns, even when they believe they are operating anonymously. This concern is particularly acute for users in high-risk environments, such as journalists, activists, or individuals living under authoritarian regimes.

We will explore these questions empirically in Chapter 4, examining whether the `auth_key_id` does indeed remain constant across sessions, how frequently (if ever) it changes, and whether it can be observed and extracted from network traffic when different transport protocols are employed.

3.1.2.2 | Perfect Forward Secrecy and `auth_key_id`

A natural question that arises is whether MTProto's PFS feature [8] mitigates the privacy concerns associated with `auth_key_id` visibility. The answer, unfortunately, is that PFS does not meaningfully address the tracking potential of observable `auth_key_id` values in network traffic.

Telegram's implementation of PFS operates by generating temporary authorization keys that are bound to a permanent authorization key. The client creates both a permanent authorization key (using `p_q_inner_data`) and a temporary key (using `p_q_inner_data_temp`). According to Telegram's documentation:

> *In order to achieve PFS, the client must never use the permanent `auth_key_id` directly. Every message that is sent to MTProto, must be encrypted by a `temp_auth_key_id`, that was bound to the `perm_auth_key_id`.*
>
> — Telegram Documentation [8]

This means that when PFS is enabled, the `auth_key_id` field that appears in message headers corresponds to the temporary key rather than the permanent key. These temporary keys are typically valid for 24 hours before expiring, at which point a new temporary key must be generated and bound to the permanent key.

However, this mechanism does not prevent tracking for several critical reasons. First, temporary keys remain valid and unchanged for their entire lifetime (typically 24 hours), during which the temporary `auth_key_id` remains constant and observable. This provides a substantial window for tracking a device's communications.

Second, and more importantly, the transition from one temporary key to another is easily observable by network adversaries. When a new temporary authorization key is negotiated, it appears in the network traffic immediately following the use of the old key. The binding process itself creates a traceable linkage: the old temporary `auth_key_id` is used to negotiate and bind the new one, and both appear in traffic from the same client, typically from the same IP address or within a brief time window. As correctly noted in analyses of MTProto's PFS implementation:

> *It is extremely unlikely for the IP address of the client and the temporary `auth_key_id` to both change at the exact same time.*
>
> — Michał "rysiek" Woźniak [5]

This creates a trivial linkability chain: if a network observer sees temporary `auth_key_id` A being used, then shortly afterward sees temporary `auth_key_id` B appear

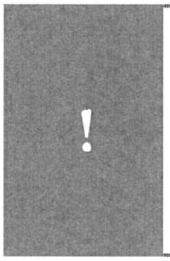
from the same IP address (or within a temporal pattern consistent with key renewal), the observer can conclude with high confidence that both keys belong to the same device. By maintaining a database of these transitions, a passive network observer can effectively track a device across arbitrary numbers of temporary key rotations, even across IP address changes and network switches.

The fundamental problem is that PFS is designed to protect against *post factum* decryption of message contents if keys are later compromised—it is not designed to prevent real-time traffic analysis or device tracking. PFS ensures that compromise of a current key does not allow decryption of past messages, but it does nothing to hide the persistent patterns and identifiers visible in unencrypted transport-layer metadata.

Therefore, while PFS provides important cryptographic guarantees about message content confidentiality, it does not address the privacy concerns associated with `auth_key_id` visibility when MTProto operates over unencrypted transports. The tracking potential remains essentially unchanged: temporary `auth_key_id` values are just as observable, just as persistent within their validity periods, and just as linkable across renewals as permanent keys would be.

3.2 | Assessing auth_key_id Exposure

Given the analysis above—which establishes that auth_key_id values are indeed persistent, observable identifiers capable of enabling device tracking by passive network observers, even in the presence of PFS—the central question becomes:



The Central Question

Which transport does Telegram actually use for MTProto?
Does Telegram default to unencrypted TCP connections, thereby exposing the auth_key_id to any network intermediary, or does it use HTTPS, which would encrypt the entire MTProto message (including the auth_key_id header) at the transport layer and protect it from passive observation?

This fundamental, crucial choice determines whether the tracking capability enabled by auth_key_id visibility is merely theoretical or represents a real-world privacy vulnerability affecting Telegram users. In the following sections, we examine Telegram's transport protocol selection across different client platforms to answer this critical question.

3.2.1 | Telegram for Mobile (Android)

Independent security researcher Michał "rysiek" Woźniak[2] conducted a detailed technical analysis of Telegram for Android, examining network traffic patterns and protocol behavior to assess auth_key_id exposure in real-world usage. [5] His investigation involved setting up a controlled testing environment using QubesOS with packet capture capabilities, analyzing Telegram for Android version v11.9.2 (5901) over multiple sessions spanning several weeks from geographically distant locations (Iceland and Poland).

Woźniak's methodology involved capturing network traffic during various Telegram activities: initial registration, background sessions, channel joining, bot interactions, and reconnections after extended periods. He encountered MTProto's obfuscation layer—a trivial scheme documented by Telegram as being designed solely to thwart simple packet filtering rather than provide security—and developed deobfuscation tooling to extract auth_key_id values from captured packets.

His findings confirmed several critical observations about auth_key_id behavior:

1. The long-term auth_key_id remains constant across sessions, IP address changes, network switches, and even when routing through Tor.

[2]Note: Mr. Woźniak formerly served as OCCRP's Head of Infrastructure and Information Security. This potential conflict of interest was properly disclosed in OCCRP's initial article [1], and Mr. Woźniak's technical claims were independently verified to be correct regardless of this potential conflict of interest. The political claims and personal opinions made in Mr. Woźniak's technical analysis were disregarded during our independent verification.

2. When PFS is enabled and temporary authorization keys are used, the temporary `auth_key_id` values appear in cleartext in network traffic.
3. Temporary keys are easily linkable to the permanent key and to each other through observable negotiation patterns: old temporary `auth_key_id` values appear in traffic immediately before new ones, typically from the same IP address.
4. Even after weeks-long breaks, previously observed temporary `auth_key_id` values reappear in traffic before new temporary keys are negotiated, creating an unbroken chain of linkability.
5. When the client connects to different server IP addresses within the same Telegram datacenter, the same `auth_key_id` continues to be used.

Woźniak's analysis demonstrated that MTProto operates over unencrypted TCP connections in practice, with all observed traffic using obfuscated TCP rather than HTTPS. This means the `auth_key_id`—whether permanent or temporary—is transmitted in cleartext (or at best trivially obfuscated) and is therefore visible to any network observer capable of passive traffic monitoring.

We independently replicated Woźniak's testing methodology and findings throughout our own investigation, confirming the correctness of his technical analysis across multiple dimensions: the persistence of `auth_key_id` values, their visibility in network traffic, the linkability of temporary keys, and the predominant use of unencrypted TCP transport rather than HTTPS.

3.2.2 | Telegram for Desktop (macOS)

To complement our replication of Woźniak's findings on Telegram for Android, we conducted an independent analysis of Telegram Desktop running on macOS to determine whether the desktop client exhibits similar `auth_key_id` exposure characteristics or employs different transport security measures.

Note: Telegram for Desktop (macOS) shares the same common codebase [9] as Telegram for Desktop (Windows) and Telegram for Desktop (Linux), making our analysis here almost certainly equally applicable to those versions of Telegram for Desktop as well.

Our testing environment consisted of a macOS system running the latest version of Telegram Desktop (version 11.15.275369 at the time of testing). We employed packet capture tooling using Wireshark and tcpdump to monitor all network traffic generated by the Telegram Desktop application during various usage scenarios. To ensure comprehensive coverage, we analyzed traffic during:

- Initial application launch and authentication

- Active messaging sessions with individual contacts and groups
- Channel browsing and subscription activities
- Background idle periods with the application remaining open
- Reconnection after network changes and application restarts

We captured traffic over multiple days of regular usage, examining packet headers, destination ports, and the structure of encrypted payloads. Additionally, we performed controlled network manipulation experiments, selectively blocking different types of traffic (plain TCP on common Telegram ports, HTTPS traffic to Telegram server IP ranges) to observe the application's fallback behavior and transport protocol preferences.

3.3 | Findings

Our analysis of Telegram Desktop on macOS revealed network behavior that initially appeared more secure than the Android client, but ultimately exposed the same fundamental vulnerability regarding `auth_key_id` visibility.

During our packet capture sessions, we observed that both Telegram for Android and Telegram for Desktop consistently establish connections to Telegram server infrastructure on port 443—the standard port for HTTPS traffic. This initially suggested that the desktop client might be using HTTPS as its transport protocol, which would provide transport-layer encryption and protect the `auth_key_id` from passive network observation.

However, detailed examination of the captured packets revealed a critical distinction: while Telegram Desktop does indeed communicate over port 443, it is *not* using HTTPS or any form of TLS encryption. Instead, the desktop client transmits MTProto payloads over plain TCP connections, exactly as observed with Telegram for Android, merely using port 443 rather than the other common Telegram ports (80, 5222, etc.).

This choice of port 443 appears to be a strategic decision to evade simplistic firewall rules and censorship mechanisms that might block non-standard ports while allowing standard HTTPS traffic. However, the traffic itself bears no resemblance to actual HTTPS: there is no TLS handshake, no certificate exchange, and no transport-layer encryption beyond MTProto's own cryptographic layer. The packets contain the characteristic structure of MTProto messages with their external headers exposed.

To verify this observation, we employed several validation techniques:

1. **TLS fingerprinting:** We analyzed the initial packets of each connection for TLS ClientHello messages. No TLS handshake signatures were present. The connections begin immediately with MTProto protocol exchanges rather than TLS negotiation.

2. **Certificate validation:** We attempted to extract TLS certificates from the captured sessions using standard TLS analysis tools. No certificates were present or exchanged at any point in the connections.

3. **Packet structure analysis:** We examined the byte-level structure of captured packets. The packets contain the MTProto external header structure—including the 64-bit `auth_key_id` and 128-bit message key—beginning immediately after the TCP header, with no intervening TLS record layer.

4. **Selective traffic blocking:** We configured local firewall rules to block outbound HTTPS traffic while permitting plain TCP on port 443. Telegram continued to function normally, confirming that it does not depend on TLS.

Most significantly, we successfully extracted `auth_key_id` values from the captured traffic using the same deobfuscation techniques applied to the Android client. The `auth_key_id` appears in cleartext (or trivially obfuscated) in the external header of every MTProto message, positioned at the expected offset immediately following the TCP payload boundary.

Our extended monitoring confirmed the same `auth_key_id` persistence characteristics observed on Android:

- The same `auth_key_id` values persisted across multiple application restarts and days of testing
- The `auth_key_id` remained constant when the system's IP address changed (switching between different WiFi networks, connecting via VPN)
- When PFS temporary keys were observed, the transitions between temporary `auth_key_id` values exhibited the same linkability patterns documented by Woźniak, with old and new temporary keys appearing in succession from the same client
- The `auth_key_id` did not change when the client connected to different Telegram server IP addresses within the same datacenter

We also observed that MTProto's obfuscation layer is applied to both clients' traffic, consistent with the Android implementation. This obfuscation involves XORing the payload with a pseudo-random stream derived from a simple initialization procedure. As documented by Telegram and confirmed by our analysis, this obfuscation provides no cryptographic security and is designed solely to prevent naive protocol detection by deep packet inspection systems looking for fixed magic bytes or protocol signatures.

The obfuscation can be reversed with trivial computational effort given knowledge of the obfuscation algorithm (which is publicly documented). Once deobfuscated, the MTProto message structure—including the plaintext `auth_key_id`—is immediately visible. We successfully developed tooling to automatically deobfuscate captured packets and extract `auth_key_id` values, confirming that the obfuscation presents no meaningful barrier to `auth_key_id` extraction by network observers.

In summary, Telegram Desktop on macOS exhibits identical `auth_key_id` exposure characteristics to Telegram for Android. Despite using port 443, which might suggest HTTPS usage to casual observers or simple traffic classification systems, the desktop client transmits MTProto over unencrypted TCP connections. The `auth_key_id` is fully visible to passive network observers positioned anywhere along the network path between client and server.

3.4 | Implications of Findings

The confirmation that both Telegram's mobile and desktop clients expose `auth_key_id` values through unencrypted transport layers has significant implications for user privacy across multiple dimensions.

Universal vulnerability across platforms: The consistent behavior across Android and macOS clients suggests that `auth_key_id` exposure is not an isolated implementation oversight in a single client, but rather reflects a systematic architectural decision in Telegram's MTProto implementation. This implies that users cannot protect themselves from `auth_key_id`-based tracking simply by switching to a different client platform. The vulnerability likely extends to all official Telegram clients across iOS, Windows, Linux, and web platforms, though we did not independently verify each platform in this study. We again note that Telegram for Desktop (macOS) shares the same common codebase [9] as Telegram for Desktop (Windows) and Telegram for Desktop (Linux), making our analysis here almost certainly equally applicable to those versions of Telegram for Desktop as well.

Deceptive use of port 443: The desktop client's use of port 443 without actual HTTPS encryption represents a particularly concerning design choice. Port 443 is universally associated with encrypted HTTPS traffic in the public consciousness and in standard security documentation. Many users, administrators, and even security professionals might reasonably assume that traffic on port 443 is encrypted at the transport layer. This assumption could lead to a false sense of security, where stakeholders believe that Telegram communications are protected from network observation when they are not.

This deceptive appearance extends to automated security tools and traffic analysis systems. Many network monitoring solutions, intrusion detection systems, and security analytics platforms classify traffic by port number. Traffic on port 443 is routinely categorized as "encrypted" or "secure" without deeper inspection. This could cause security monitoring systems to miss or misclassify the privacy implications of Telegram traffic, failing to alert on the presence of persistent identifiers in supposedly encrypted channels.

Scope of adversary visibility: The use of unencrypted TCP transport means that `auth_key_id` values are visible to an extremely broad range of potential adversaries. This includes:

- **Internet Service Providers:** All ISPs along the path between user and Telegram servers could technically observe and log `auth_key_id` values via active packet capture, subject to legal authorization.
- **Network administrators:** Corporate networks, university networks, coffee shop WiFi operators, and any other network through which Telegram traffic passes can extract `auth_key_id` values.

- **Government surveillance:** State-level actors conducting mass surveillance or targeted monitoring can collect `auth_key_id` values at internet exchange points, through compelled ISP cooperation, or via direct network taps.
- **Malicious hotspot operators:** Attackers operating rogue WiFi access points can harvest `auth_key_id` values from users who connect through compromised networks.
- **Passive eavesdroppers:** Any party capable of passive traffic monitoring—whether through physical access to network infrastructure, wireless interception, or other means—can extract `auth_key_id` values without any active attack or protocol manipulation.

Notably, all of these adversaries can collect `auth_key_id` values through entirely passive observation. No man-in-the-middle attack is required, no certificate compromise is necessary, and no active protocol manipulation is needed. Simple packet capture and trivial deobfuscation suffice to extract persistent device identifiers from Telegram traffic.

Long-term tracking capability: The persistence of `auth_key_id` values across sessions, network changes, and extended time periods enables long-term tracking of individual devices. An adversary who collects `auth_key_id` values from network traffic can build a comprehensive database associating specific `auth_key_id` values with observed network locations, timestamps, traffic patterns, and—when the user's identity is known through other means—specific individuals.

Such a database could enable queries like: "Where has the device associated with `auth_key_id` X been observed over the past six months?" or "Which `auth_key_id` values were present at network location Y during time period Z?" The ability to track devices across IP address changes means that even users who employ dynamic IP addressing, regularly switch networks, or use mobile connections remain trackable through their persistent `auth_key_id`.

Circumvention of anonymity tools: The `auth_key_id` exposure has particularly severe implications for users attempting to achieve anonymity through tools like VPNs or Tor. Even when a user routes their Telegram traffic through such anonymity networks, the persistent `auth_key_id` provides a stable identifier that can be correlated across different anonymity sessions.

For example, if a user connects to Telegram through Tor, the exit node operators and any downstream network observers can still extract the `auth_key_id`. If the same user later connects without Tor, or through a different Tor circuit, the reappearance of the same `auth_key_id` immediately links the supposedly anonymous and non-anonymous sessions. The anonymity provided by the network-layer tools is undermined by the application-layer persistent identifier.

This is particularly concerning given Telegram's popularity among journalists, activists, and dissidents who may face significant personal risk if their communications can be tracked. Users in such high-risk categories often employ sophisticated operational security measures, including regular network switching, VPN usage, and careful compartmentalization of identities. However, the persistent `auth_key_id` can defeat many of these countermeasures if network traffic is observable.

Ineffectiveness of PFS against tracking: As discussed earlier, MTProto's PFS implementation does not mitigate `auth_key_id`-based tracking. Our observations confirm that temporary authorization keys, while providing protection against post-compromise decryption, are just as visible and linkable as permanent keys. The 24-hour validity period of temporary keys provides substantial tracking windows, and the observable transitions between temporary keys allow indefinite tracking across key rotations.

This means that users cannot protect themselves from `auth_key_id` tracking by enabling PFS or by any configuration changes within Telegram. The vulnerability is inherent to the protocol's transport layer choices rather than its cryptographic mechanisms.

Contrast with competing platforms: The `auth_key_id` exposure stands in stark contrast to the transport security approaches employed by Telegram's competitors. WhatsApp encrypts its Noise Protocol transport within TLS, ensuring that session identifiers and protocol metadata are protected from passive observation. Signal similarly uses TLS as its transport layer. These design choices reflect a defense-in-depth philosophy where transport-layer encryption protects against traffic analysis even if application-layer encryption were somehow compromised.

Telegram's decision to forego transport-layer encryption—despite having documented support for HTTPS transport in the MTProto specification—represents a meaningful divergence from what has become standard practice in privacy-focused messaging applications. The question of why Telegram chose to default to unencrypted transports, despite the clear privacy implications and the availability of encrypted alternatives, remains unanswered by public documentation.

Challenges for threat modeling: The `auth_key_id` exposure complicates threat modeling for organizations and individuals considering Telegram for sensitive communications. Standard threat models for encrypted messaging applications typically assume that while message metadata (such as sender, recipient, and timestamp) might be visible to service providers, such metadata is not visible to network intermediaries due to transport-layer encryption.

Telegram's architecture violates this assumption by exposing persistent device identifiers to the entire network path. This requires security professionals to account for a much broader adversary model when assessing Telegram's suitability for particular use cases. Any threat model that includes passive network observation as a relevant

adversary capability must now account for the possibility of device tracking through `auth_key_id` collection.

Regulatory and compliance implications: In jurisdictions with data protection regulations such as the General Data Protection Regulation (GDPR), the exposure of persistent device identifiers may have compliance implications. Device identifiers that can be used to single out and track individuals may constitute personal data subject to regulatory requirements. Organizations that deploy or recommend Telegram must consider whether the `auth_key_id` exposure creates data protection obligations or risks, particularly if network traffic logs containing `auth_key_id` values are collected and retained by ISPs or network operators.

Remediation challenges: From a remediation perspective, addressing the `auth_key_id` exposure requires Telegram to enforce HTTPS (or more precisely, TLS) as the mandatory and exclusive transport method for MTProto, removing support for all unencrypted transport options including plain TCP, WebSocket over plain HTTP, and unencrypted HTTP transports. This would require coordinated updates across all client platforms to remove code paths that implement or fall back to unencrypted transports, and corresponding server-side configuration changes to reject connections that do not use TLS. While this approach would impose additional server-side computational costs for TLS termination at scale, these costs are a necessary and proportionate investment in user privacy given the severity of the tracking capability enabled by the current architecture.

Alternatively, Telegram could implement more frequent authorization key rotation, ensuring that `auth_key_id` values change regularly and automatically. However, this would not eliminate tracking within the validity period of each key and would still allow linkability through observable key transitions unless the rotation mechanism were carefully designed to prevent timing correlation.

In conclusion, the cross-platform nature of `auth_key_id` exposure through unencrypted transports represents a systematic privacy vulnerability affecting all Telegram users. The scope of potential adversaries, the persistence of the tracking capability, and the ineffectiveness of user-level countermeasures combine to create a significant gap between Telegram's public privacy positioning and the actual privacy properties delivered by its protocol implementation. Users who rely on Telegram for sensitive communications should be aware that their device identifiers are visible to passive network observers, enabling long-term tracking that persists across network changes, anonymity tool usage, and geographic relocation.

4

Critical Evaluation

Having established in Chapter 3 the technical architecture of MTProto, the nature and persistence of `auth_key_id` values, and the empirical finding that both Telegram for Android and Telegram for Desktop transmit MTProto over unencrypted TCP connections (thereby exposing `auth_key_id` to passive network observers), we now turn to evaluating specific claims made in public reporting about the privacy implications of these design choices.

The OCCRP published an investigative article [1] examining Telegram's security architecture and network infrastructure, with particular focus on the privacy implications of `auth_key_id` exposure and the potential for surveillance by parties with access to network infrastructure. The article makes several specific technical and privacy-related claims that warrant careful evaluation.

Our objective in this chapter is to assess these claims with judicial impartiality—evaluating each assertion based on the technical evidence established in our independent analysis, examining both the factual accuracy of the claims and the reasonableness of the inferences drawn from technical facts. Where claims are supported by our findings, we acknowledge this; where claims overreach or introduce speculative elements not directly supported by technical evidence, we note these limitations as well.

This evaluation is structured around five principal claims extracted from the OCCRP investigation. For each claim, we present the assertion as stated in the original reporting, examine the technical premises underlying the claim, assess the claim's validity against our empirical findings, and discuss any qualifications, limitations, or contextual factors that bear on a fair evaluation of the claim's accuracy and significance.

4.1 | Claim 1: auth_key_id Enables User Tracking

The OCCRP investigation states:

> *The unencrypted part is called 'auth_key_id.' This makes it possible to identify a specific user device. [...] If I know your device's 'auth_key_id,' and I can listen in on the network that handles the data... I know it is your specific device communicating with Telegram servers.*
>
> — **OCCRP** quoting internal security expert Michał "rysiek" Woźniak

This claim suggests that the auth_key_id, transmitted in plaintext as part of the MTProto protocol, serves as a persistent device identifier that could be used to track users across sessions and locations.

4.1.1 | Technical Premises

This claim rests on three technical premises:

1. That the auth_key_id is transmitted in an unencrypted (or only trivially obfuscated) form accessible to network observers
2. That the auth_key_id uniquely identifies a specific device across multiple communication sessions
3. That observation of auth_key_id values in network traffic is sufficient to track device activity

4.1.2 | Evaluation Against Evidence

Premise 1 (Unencrypted transmission): Our independent analysis in Chapter 3 confirms that both Telegram for Android and Telegram for Desktop transmit MTProto over unencrypted TCP connections. While MTProto applies a trivial obfuscation layer to its payloads, we successfully deobfuscated captured traffic and extracted auth_key_id values using publicly documented techniques. The auth_key_id appears in the external header of every MTProto message and is accessible to any party capable of passive packet capture along the network path. This premise is **factually accurate**.

Premise 2 (Device-specific persistence): Our extended monitoring of Telegram clients confirmed that `auth_key_id` values remain constant across multiple dimensions: application restarts, IP address changes, network switches, connections to different Telegram server IP addresses, and time periods spanning days and weeks. When PFS temporary keys are used, we observed that temporary `auth_key_id` values persist for their entire validity period (typically 24 hours) and that transitions between temporary keys are readily observable and linkable. These findings replicate those documented by Woźniak [5] and confirm that `auth_key_id` values function as persistent device identifiers. This premise is **factually accurate.**

Premise 3 (Tracking capability): Given that `auth_key_id` values are both observable in network traffic and persistent across sessions, the technical capability for device tracking follows directly. A passive network observer positioned at any point along the network path—including ISPs, network administrators, or state-level surveillance actors—can extract `auth_key_id` values from captured packets, maintain a database associating `auth_key_id` values with timestamps and network locations, and query this database to track device activity across IP address changes and geographic locations. This premise is **technically sound.**

4.1.3 | Assessment

Claim 1 is **substantiated by technical evidence.** The `auth_key_id` is indeed transmitted without transport-layer encryption in standard Telegram client configurations, it does uniquely identify specific devices with high persistence, and passive network observation is sufficient to enable tracking of device communications with Telegram servers.

The claim is narrowly scoped and makes no assertions beyond what the technical evidence directly supports. It correctly notes that this tracking capability depends on the adversary's ability to "listen in on the network that handles the data," appropriately identifying the adversary model (passive network observation) required for exploitation.

Limitations and qualifications: While the claim is technically accurate, several contextual factors warrant mention for completeness:

- The claim addresses device tracking, not necessarily user identity tracking. An observer can determine that "device with `auth_key_id` X is active" but cannot directly infer the human user's identity from the `auth_key_id` alone unless that identifier has been previously associated with a known individual through other means.
- The tracking capability depends on the adversary's position on the network path. Adversaries without access to network infrastructure along the client-to-server route cannot observe `auth_key_id` values. However, given the breadth

of parties with such access (ISPs, corporate network administrators, government surveillance programs, etc.), this limitation is not particularly restrictive in practice.

- The claim does not address whether this tracking capability is intentional, a design oversight, or a deliberate trade-off made for other engineering reasons. The technical fact of vulnerability is established, but the claim makes no assertion about intent or negligence.

These qualifications do not undermine the claim's validity but rather clarify its precise scope and the nature of the privacy risk it identifies.

4.2 | Claim 2: Metadata Collection

According to the OCCRP investigation:

> *By looking at the network packets... I also get your IP address at a given time, which tells me your rough geographic location. [...] In such an attack, the hackers aren't even interested so much in the user's correspondence. They get metadata to analyze. And that means IP addresses, user locations, who exchanges data packets with whom, the kind of data it is... really, all possible information.*
>
> — **OCCRP** quoting internal security expert Michał "rysiek" Woźniak

4.2.1 | Technical Premises

This claim involves several distinct technical assertions:

1. That network-level observation reveals IP addresses and timing information
2. That IP addresses provide approximate geographic location information
3. That network observers can determine communication patterns ("who exchanges data packets with whom")
4. That network observers can characterize traffic types ("the kind of data it is")
5. That this metadata collection does not require access to message content

4.2.2 | Evaluation Against Evidence

Premise 1 (IP address and timing visibility): This is a fundamental property of IP networking and is not specific to Telegram. Any network observer positioned on the communication path necessarily observes source and destination IP addresses and packet timing in order to route traffic. This applies universally to all internet communications unless specifically protected by additional layers of anonymization (such as Tor or VPNs). This premise is **trivially accurate** but not unique to Telegram. However, **the OCCRP article's presentation could be considered misleading** in that it emphasizes this property as though it were a unique vulnerability of Telegram, when

in fact IP address and timing visibility are inherent characteristics of all unencrypted internet traffic.

Premise 2 (IP geolocation): IP addresses can indeed be correlated with approximate geographic locations using commercial geolocation databases, typically providing accuracy at the city or regional level for most IP addresses. The precision varies based on the type of connection, with mobile IP addresses generally providing coarser location information than fixed broadband connections. This is a well-established capability in network analysis and threat intelligence. This premise is **factually accurate**. However, it is crucial to emphasize that **IP address visibility is not unique to Telegram whatsoever**—Signal, WhatsApp, and virtually all internet applications transmit IP addresses as a fundamental requirement of IP networking. Presenting this as a Telegram-specific vulnerability, as the article's framing suggests, is **actively misleading** and misrepresents the nature of the issue.

Premise 3 (Communication pattern analysis): The claim that network observers can determine "who exchanges data packets with whom" requires careful parsing. Network observers can certainly see that device with `auth_key_id` X is communicating with Telegram server IP address Y. However, because Telegram uses a centralized server architecture (rather than peer-to-peer communication), network observers positioned between client and server typically cannot directly observe which specific Telegram users are communicating with each other merely from network traffic patterns.

The server-side communication graph—which users are messaging which other users—is only visible to Telegram's servers themselves or to parties who have compromised those servers. A passive network observer between client and server sees traffic flowing to and from Telegram's infrastructure but cannot typically determine the ultimate recipients of messages without additional information or traffic correlation capabilities.

However, timing analysis and traffic pattern analysis might reveal some information about communication patterns. For example, if two users' devices consistently show traffic bursts at correlated times, this might suggest mutual communication, though this inference would be probabilistic rather than certain. Similarly, the volume and frequency of traffic might reveal some information about usage patterns even if specific recipients cannot be identified.

This premise is therefore **partially accurate**: network observers gain substantial metadata visibility, but the claim overstates the degree to which communication graphs can be reconstructed through network observation alone in a centralized server architecture. This is an important distinction.

Premise 4 (Traffic type characterization): The ability to determine "the kind of data it is" depends on what level of granularity is intended. At a coarse level, network observers can distinguish between different types of Telegram activity through traffic analysis:

- Voice/video calls typically produce continuous streams of relatively uniform packet sizes at regular intervals
- Text messaging produces small, bursty traffic patterns
- File transfers produce large, sustained data flows
- Media downloads produce substantial inbound traffic bursts

These traffic patterns can be characterized through statistical analysis even when message content is encrypted. Traffic analysis is a well-established technique in network security and surveillance, and MTProto's encryption does not prevent inference of activity types based on timing, volume, and flow characteristics.

However, the claim's phrase "all possible information" represents an overstatement. While substantial metadata can be extracted through traffic analysis, network observers without access to Telegram's servers cannot determine specific message content, contact lists, group memberships (beyond statistical inference), or many other aspects of user activity. The phrase "all possible information" suggests a degree of visibility that exceeds what network-level observation alone provides.

This premise is **substantially accurate regarding traffic characterization capabilities** but **overstated in its formulation as "all possible information."**

Premise 5 (Metadata collection without content access): This premise is unambiguously correct. All of the metadata discussed—IP addresses, timing, traffic patterns, statistical characteristics—is visible in packet headers and flow characteristics without any need to decrypt message content. This is precisely the distinction between traffic analysis (which operates on observable network metadata) and content analysis (which requires decryption). This premise is **factually accurate.**

4.2.3 | Assessment

Claim 2 is **substantially accurate but contains elements of overstatement, and at times appears to be actively misleading the audience.** The core assertion—that network-level access enables significant metadata surveillance providing IP addresses, location information, timing data, and traffic characterization—is supported by both general principles of network analysis and our specific findings regarding Telegram's unencrypted transport.

However, three aspects of the claim warrant qualification:

1. The statement that network observers can determine "who exchanges data packets with whom" overstates the visibility provided by client-to-server network observation in a centralized architecture. Network observers can see that specific devices are communicating with Telegram's servers, but cannot directly

reconstruct the server-side communication graph without additional capabilities or access.

2. The phrase "all possible information" represents rhetorical overreach. While the metadata visibility is substantial and privacy-significant, it does not constitute literally all information about user activity. Message content, contact lists, group memberships, and many other elements of user activity remain opaque to passive network observers.

3. **Most critically, the metadata surveillance capabilities described in this claim are not unique to Telegram whatsoever.** Signal, WhatsApp, and virtually all internet-based messaging applications expose the same fundamental metadata to network observers: IP addresses, timing patterns, traffic volumes, and statistical characteristics. These are inherent properties of IP networking that apply universally unless specifically mitigated through additional anonymization layers (such as Tor or VPNs). The OCCRP article's framing presents these metadata surveillance capabilities as though they were Telegram-specific vulnerabilities, which is **actively misleading.** While the `auth_key_id` exposure discussed in Claim 1 is indeed specific to Telegram's protocol design, the broader metadata surveillance capabilities described here affect all messaging platforms equally. Singling out Telegram for criticism on this point, without acknowledging that Signal and WhatsApp face identical metadata exposure, fundamentally misrepresents the nature and scope of the issue.

These qualifications address the scope and precision of the claim rather than its fundamental validity. The underlying technical point—that unencrypted transport exposes substantial metadata to network surveillance—is sound. The concern about metadata surveillance is legitimate and technically grounded. The overstatements appear to be rhetorical emphasis rather than fundamental technical errors, but they do reduce the precision of the claim.

A more precisely formulated version might state: "Network-level observers can extract substantial metadata including IP addresses, approximate location, timing patterns, and statistical traffic characteristics, enabling surveillance of device activity and communication patterns even without access to message content." This formulation would preserve the claim's essential point while avoiding overstatement.

4.3 | Claim 3: E2EE Does Not Prevent Tracking

The OCCRP investigation makes the following assertion:

> *Even users who use its 'end-to-end' encryption features are vulnerable to being tracked by anyone who can monitor its network traffic.*
>
> — OCCRP

This claim challenges the fundamental security assumptions of Telegram's Secret Chat feature, which purports to provide complete privacy through end-to-end encryption.

4.3.1 | Technical Premises

This claim rests on the premise that end-to-end encryption, as implemented in Telegram's Secret Chats, does not prevent the tracking vulnerability identified in Claim 1, because the `auth_key_id` exposure occurs at the MTProto transport layer, which underlies the Secret Chat encryption layer.

4.3.2 | Evaluation Against Evidence

To evaluate this claim, we must understand the relationship between Telegram's Secret Chats (which provide end-to-end encryption) and the MTProto transport layer (which provides server-client encryption for regular "Cloud Chats").

Secret Chats implement end-to-end encryption *over* the MTProto transport layer. The architecture is layered as follows:

1. **Application layer:** Secret Chat messages are encrypted end-to-end using a separate encryption protocol before transmission
2. **MTProto layer:** The already-encrypted Secret Chat messages are then packaged within MTProto's server-client encryption
3. **Transport layer:** The MTProto messages (containing the encrypted Secret Chat content) are transmitted over TCP, with the MTProto external header (including `auth_key_id`) in cleartext

Crucially, the `auth_key_id` is part of the MTProto external header, which is transmitted regardless of whether the MTProto payload contains regular Cloud Chat messages or end-to-end encrypted Secret Chat messages. The transport-layer tracking vulnerability identified in Claim 1 operates at the MTProto header level, not the message content level.

Therefore, using Secret Chats protects the content of messages from visibility by Telegram's servers (since the messages are end-to-end encrypted), but it does *not* prevent network observers from extracting `auth_key_id` values from MTProto headers and using those identifiers to track device activity.

Our empirical analysis did not specifically test Secret Chat traffic (as Secret Chats are only available in mobile clients and our detailed packet captures focused on regular messaging). However, the architectural relationship between Secret Chats and MTProto transport is documented in Telegram's technical specifications [2] and is consistent with the layered protocol design.

The claim's core premise—that end-to-end encryption does not prevent tracking via `auth_key_id` observation—is therefore **architecturally sound**. The tracking occurs at a protocol layer beneath the end-to-end encryption layer, rendering the end-to-end encryption irrelevant to this particular vulnerability.

4.3.3 | Assessment

Claim 3 is **technically accurate** with respect to `auth_key_id`-based device tracking. End-to-end encryption in Secret Chats protects message content from observation by Telegram's servers and by network intermediaries, but it does not protect against device tracking through `auth_key_id` observation because the `auth_key_id` appears in MTProto headers that are transmitted regardless of message content encryption.

Important contextual clarifications:

1. The claim is specifically about *tracking* (determining that a specific device is active and communicating with Telegram), not about *content access* (reading message contents). End-to-end encryption successfully protects content in Secret Chats; the claim correctly notes that content protection does not prevent device tracking.

2. The claim applies specifically to `auth_key_id`-based tracking through network observation. Secret Chats do provide other privacy benefits beyond content encryption, such as forward secrecy, self-destructing messages, and prevention of server-side message storage. The claim does not negate these benefits; it identifies a specific limitation regarding device tracking.

3. The phrase "vulnerable to being tracked by anyone who can monitor its network traffic" is precise and appropriate. It correctly identifies the adversary ca-

pability required (network traffic monitoring) and the nature of the vulnerability (tracking, not content compromise).

This claim is particularly significant because it addresses a potential misconception. Users who enable Secret Chats might reasonably believe they have maximized their privacy protections and are immune to surveillance by network observers. The claim correctly identifies that this assumption is false with respect to device tracking, even though it remains true with respect to content protection.

The claim does not overstate its scope or make assertions beyond what the technical architecture supports. It is a precise, accurate statement about a specific limitation of Secret Chats' privacy protections against a specific class of adversary (network observers).

4.4 | Claim 4: Infrastructure Control Enables Surveillance

The OCCRP article states:

> *If someone has access to Telegram traffic and cooperates with Russian intelligence services, this means that the device identifier becomes a really big problem — a tool for global surveillance of messenger users, regardless of where they are and what server they connect to.*
>
> — **OCCRP** quoting internal security expert Michał "rysiek" Woźniak

This claim suggests that control over Telegram's network infrastructure could enable comprehensive, global tracking of users.

4.4.1 | Technical Premises

This claim involves several technical and geopolitical premises:

1. That the `auth_key_id`-based tracking capability, if exploited at scale through infrastructure access, could enable comprehensive surveillance
2. That such surveillance would function "regardless of where [users] are," implying global reach
3. That such surveillance would function regardless of "what server they connect to," implying persistence across Telegram's distributed infrastructure
4. That cooperation with state intelligence services (specifically Russian intelligence) is relevant to the threat model

4.4.2 | Evaluation Against Evidence

Premise 1 (Scaled surveillance capability): Our analysis confirms that `auth_key_id` values are persistent device identifiers visible to network observers. If an entity controls network infrastructure through which Telegram traffic flows—such as internet exchange points, ISP infrastructure, or routing equipment—they could systematically collect `auth_key_id` values from all passing Telegram traffic, associate these identifiers with timestamps, IP addresses, and traffic characteristics, and build a comprehensive database enabling device tracking.

The technical capability for such scaled surveillance follows directly from the technical facts established in Claims 1-3. The phrase "really big problem" is a subjective characterization rather than a technical claim, but the underlying technical capability is sound. This premise is **technically accurate**.

Premise 2 (Global reach): The assertion that such surveillance would function "regardless of where they are" requires examination. The `auth_key_id` remains constant when users change geographic locations, as confirmed by our testing across different networks and by Woźniak's testing across different countries (Iceland and Poland). Therefore, from a user mobility perspective, the claim is accurate: the same device maintains the same `auth_key_id` regardless of geographic location.

However, from a surveillance infrastructure perspective, "global" reach depends on the extent of the adversary's network access. An entity with control over infrastructure in one geographic region can only observe traffic passing through that infrastructure. Achieving truly global surveillance would require infrastructure access across multiple geographic regions and network paths.

The claim's invocation of "Russian intelligence services" and the geopolitical context of the broader OCCRP investigation suggests concern about a state-level adversary with potentially extensive international infrastructure access. Whether such access exists, and to what extent, is beyond the scope of pure technical analysis and enters the domain of intelligence assessment and geopolitical analysis.

This premise is **technically accurate regarding `auth_key_id` persistence across locations** but involves **extrapolation regarding infrastructure access extent** that cannot be definitively confirmed or refuted through technical analysis alone.

Premise 3 (Persistence across servers): The claim states that surveillance would function regardless of "what server they connect to." Our testing confirmed that `auth_key_id` values remain constant when clients connect to different Telegram server IP addresses within the same datacenter. Telegram operates multiple datacenters across different geographic regions, and the MTProto protocol includes datacenter migration capabilities for load balancing and availability.

The authorization key is generated on the client device and is used consistently by that device regardless of which Telegram datacenter or server IP address it connects to. The `auth_key_id`, being derived from the authorization key, similarly remains constant across server connections.

Therefore, from the client device's perspective, the same `auth_key_id` is used regardless of backend server connections. This confirms the technical accuracy of the claim that tracking functions "regardless of what server they connect to." This premise is **technically accurate**.

Premise 4 (Intelligence service cooperation): The claim introduces a specific geopolitical scenario: "If someone has access to Telegram traffic and cooperates with Russian intelligence services." This formulation involves assumptions about adver-

sary identity, motivations, and relationships that extend beyond pure technical analysis.

From a technical perspective, the identity of the potential adversary (whether Russian intelligence services, other state actors, criminal organizations, or corporate entities) does not change the fundamental tracking capability. The `auth_key_id` exposure creates a technical vulnerability that could be exploited by any adversary with network access, regardless of their identity or motivation.

The specific invocation of Russian intelligence services appears to relate to the geopolitical context of the broader OCCRP investigation, which examined Telegram's infrastructure relationships and corporate structure. Evaluating whether specific intelligence services have access to specific network infrastructure is an intelligence and investigative question rather than a technical security question.

A more neutral technical formulation might state: "If an entity has access to network infrastructure through which Telegram traffic flows, the `auth_key_id` exposure enables comprehensive tracking of device activity." The addition of specific adversary attribution ("Russian intelligence services") introduces elements that cannot be verified through technical analysis alone.

This premise involves **technical accuracy regarding the surveillance capability** combined with **geopolitical attribution that is beyond technical scope.**

4.4.3 | Assessment

Claim 4 is a **hybrid claim combining technically accurate elements with geopolitical assumptions and threat modeling that extends beyond pure technical analysis.**

Technically sound elements:

- The `auth_key_id` exposure does enable device tracking by parties with network infrastructure access
- This tracking capability persists across user geographic locations
- This tracking capability persists across different Telegram server connections
- Systematic collection at infrastructure scale could enable comprehensive surveillance of Telegram device activity

Elements involving extrapolation or assumption:

- The extent to which any particular entity (including Russian intelligence services) actually has access to relevant network infrastructure is an empirical and investigative question, not a technical one

- The specific attribution to Russian intelligence services relates to the investigation's broader geopolitical findings rather than technical analysis

Fair assessment: The claim's technical core is sound: infrastructure-level access combined with `auth_key_id` visibility could enable extensive device tracking. However, the claim blends this technical reality with geopolitical threat modeling and specific adversary attribution.

For a reader focused solely on technical security properties, the key takeaway is validated: the architecture enables infrastructure-based surveillance regardless of who might exploit it. For a reader evaluating the claim's geopolitical dimensions (whether Russian intelligence services specifically have such access and intent), technical analysis alone cannot provide definitive answers.

Critical note on investigative methodology: The OCCRP investigation attempts to establish connections between Telegram's infrastructure providers and Russian intelligence services through circumstantial evidence. Specifically, the investigation identifies Vladimir Vedeneev (owner of Global Network Solutions, which manages some Telegram infrastructure) and his connections to GlobalNet, a Russian telecommunications provider. The investigation notes that GlobalNet has clients including GlavNIVTS (a Russian government research center with reported intelligence connections) and that another company partly owned by Vedeneev (Electrontelecom) provides services to FSB offices.

However, these connections constitute **guilt by association through multiple degrees of separation** rather than direct evidence of intelligence service access to Telegram infrastructure. The logical chain involves several inferential leaps: (1) Vedeneev's company manages some Telegram infrastructure; (2) Vedeneev founded a separate company (GlobalNet) that has government clients; (3) therefore, Russian intelligence might have access to Telegram traffic. This reasoning pattern is fundamentally speculative.

Comparable scenarios not examined: The same investigative methodology could potentially be applied to virtually any messaging platform with similar circumstantial results:

- **Signal and Amazon Web Services:** Signal's infrastructure runs on Amazon Web Services (AWS). AWS maintains extensive contracts with the U.S. Intelligence Community, including a $600 million contract with the CIA [10] and subsequent contracts with the NSA and other intelligence agencies. Amazon's CEO has publicly emphasized the company's commitment to supporting U.S. defense and intelligence work. Using the OCCRP's investigative logic, one could argue this creates potential for NSA surveillance of Signal users—yet such claims are rarely prominently featured in Western investigative journalism.

- **WhatsApp and Meta's government relationships:** WhatsApp is owned by Meta (Facebook), which has documented cooperative relationships with U.S. law enforcement and intelligence agencies, provides data in response to thousands of government requests annually, [11] and employs former intelligence officials in security positions. Meta's infrastructure providers similarly have government contracts. The same inferential chain could suggest potential for U.S. intelligence access.
- **iMessage and Apple's data centers:** Apple's iMessage service relies on data center infrastructure from multiple providers with government contracts. Apple has acknowledged compliance with government data requests and has employed former NSA personnel. Similar circumstantial connections could be drawn.
- **Microsoft Teams and Azure:** Microsoft Azure, which hosts Microsoft Teams and other communication platforms, maintains substantial intelligence community contracts and has acknowledged cooperating with government data requests. The same guilt-by-association logic would apply.

The fundamental issue is that **any major technology infrastructure provider in any country will have some degree of relationship with that country's government and security services.** This is inevitable when companies operate telecommunications infrastructure, data centers, or cloud services at scale. The existence of such relationships—whether through client relationships, government contracts, regulatory compliance, or business partnerships—does not constitute evidence of active surveillance cooperation or intelligence service access to specific user data.

The OCCRP investigation appears to apply a standard of suspicion to Russian-connected infrastructure that is not symmetrically applied to U.S.- or European-connected infrastructure, despite comparable (and often more direct) relationships between Western technology providers and Western intelligence services. This asymmetry suggests **a politically motivated framing rather than a neutral security analysis.**

The technical vulnerability remains valid: Importantly, this critique of the investigation's geopolitical framing does not diminish the technical validity of the `auth_key_id` exposure. The technical vulnerability is real and affects all Telegram users regardless of which specific adversaries might exploit it. Infrastructure providers in *any* country—whether Russia, the United States, China, or elsewhere—could potentially exploit this vulnerability if they have network access.

The appropriate framing is: "The `auth_key_id` exposure creates a technical vulnerability that could be exploited by any entity with network infrastructure access, including but not limited to state-level actors in multiple jurisdictions." This formulation acknowledges the technical risk without unsupported attribution to specific adversaries.

The claim would be more clearly separated into technical and non-technical components if formulated as: "The `auth_key_id` exposure creates a technical capability for comprehensive device tracking by any entity with infrastructure access; the OCCRP investigation suggests concern about specific actors potentially having such access, though the evidence for such access consists primarily of circumstantial business relationships rather than direct proof of surveillance capability or intent." This formulation distinguishes technical fact from investigative speculation.

4.5 | Claim 5: Router Control Enables Tracking

The OCCRP investigation states:

> *If a company controls the routers that distribute traffic passing through Telegram servers, this means that it, or anyone to whom it grants such access, can see the identifiers of messenger users.*
>
> — **OCCRP** quoting internal security expert Michał "rysiek" Woźniak

This claim implies that network-level control at specific infrastructure points could compromise user privacy even without breaking encryption.

4.5.1 | Technical Premises

This claim rests on the premise that control of routing infrastructure provides visibility into `auth_key_id` values as they transit network equipment, and that such access could be granted to third parties by the controlling entity.

4.5.2 | Evaluation Against Evidence

The technical premise is straightforward and follows directly from the findings established in previous claims. Routers and other network infrastructure equipment necessarily process packet headers in order to forward traffic to appropriate destinations. If Telegram traffic passes through routing infrastructure controlled by a particular entity, and if that traffic contains `auth_key_id` values in cleartext or trivially obfuscated form (as we have established it does), then the controlling entity has access to those identifier values.

This is not a vulnerability in the routers themselves, but rather a consequence of Telegram's protocol design choice to transmit `auth_key_id` in the MTProto external header without transport-layer encryption. Any network equipment that forwards the packets can access the header information.

The phrase "or anyone to whom it grants such access" is also technically straightforward. An entity controlling network infrastructure can implement logging, monitoring, or traffic mirroring capabilities that make packet data (including `auth_key_id`

values) available to other parties. This is a routine capability in network management and can be used for legitimate purposes (traffic analysis, security monitoring) or surveillance purposes depending on context and authorization.

Our analysis provides direct empirical support for this claim: we successfully captured and analyzed Telegram traffic, extracted `auth_key_id` values, and demonstrated that these identifiers are visible to network observers. If we could perform this analysis from an endpoint position, entities with infrastructure-level access would have even more comprehensive visibility across all traffic passing through their equipment.

4.5.3 | Assessment

Claim 5 is **technically accurate and directly supported by our empirical findings**. It is essentially a specific application of the general principle established in Claim 1, focused on the particular case of routing infrastructure control.

The claim is narrowly scoped, making no assertions beyond the technical capability for identifier visibility. It does not claim that such surveillance is occurring, only that the technical capability exists. It does not specify which companies or entities might have such infrastructure control, leaving that as an investigative question separate from the technical analysis.

The phrase "even without breaking encryption" is particularly important and accurate. This correctly identifies that the privacy compromise occurs at the transport layer rather than through cryptographic attack. The MTProto message contents remain encrypted; the vulnerability is in the exposure of metadata through protocol design rather than through encryption failure.

4.6 | Summary of Critical Evaluation

Having evaluated each of the five principal claims from the OCCRP investigation against our independent technical analysis, we can summarize our findings:

- **Claim 1 (Unencrypted `auth_key_id` enables tracking):** Fully supported by evidence. Technically accurate without overstatement.
- **Claim 2 (Network access enables metadata surveillance):** Substantially accurate with elements of overstatement. Core technical points are sound; some formulations ("who exchanges data packets with whom," "all possible information") exceed what network observation strictly provides in a centralized architecture, though the fundamental privacy concern is valid.
- **Claim 3 (End-to-end encryption does not prevent tracking):** Fully supported by evidence. Technically accurate and appropriately scoped. Correctly identifies an important limitation of Secret Chats' privacy protections.
- **Claim 4 (Infrastructure control enables global surveillance):** Technical core is accurate; claim combines sound technical analysis with geopolitical threat modeling and specific adversary attribution based on circumstantial evidence rather than direct proof. The technical capability for infrastructure-based surveillance is confirmed; whether specific actors have access and intent is an investigative rather than technical question, and the investigation's evidence consists primarily of guilt-by-association reasoning that could equally be applied to Western messaging platforms and their infrastructure providers.
- **Claim 5 (Router control provides identifier visibility):** Fully supported by evidence. Technically accurate and directly validated by our empirical findings.

Overall assessment: The OCCRP investigation's technical claims regarding `auth_key_id` exposure and tracking capability are predominantly accurate and supported by independent technical analysis. The core technical findings—that Telegram transmits MTProto over unencrypted TCP connections, that `auth_key_id` values are visible to network observers, that these identifiers are persistent and enable device tracking, and that this vulnerability affects all Telegram users including those using Secret Chats—are sound and reproducible.

Where the investigation's claims extend into geopolitical threat modeling, specific adversary attribution, or characterizations that blend technical facts with subjective risk assessment, we note these elements as beyond the scope of pure technical validation. **Importantly, the investigation's attempts to establish connections between Telegram's infrastructure and Russian intelligence services rely on circumstantial evidence and inferential reasoning that falls well short of direct proof.** The investigative methodology—connecting infrastructure providers to government clients

through multiple degrees of separation—could equally be applied to any major technology platform, yet such scrutiny appears to be selectively applied based on geopolitical orientation rather than applied consistently across platforms regardless of national origin.

Operators' routine operational logs and lawful obligations typically cover higher-level connection metadata – for instance, IP addresses, timestamps, volumes and session identifiers used for billing, routing and network management. These routine metadata do not ordinarily include transport-level payloads or internal transport headers such as `auth_key_id`. Extraction and storage of transport-level identifiers generally requires active packet capture or interception beyond ordinary operational monitoring and, absent a lawful authorisation, is restricted or prohibited under applicable law.

This does not diminish the technical validity of the `auth_key_id` vulnerability itself, which remains a legitimate privacy concern regardless of which specific actors might exploit it. The investigation performs a valuable service in bringing technical attention to a protocol design choice (unencrypted transport exposing persistent identifiers) that has significant privacy implications for Telegram's user base. The technical analysis underlying the investigation is competent and, where we were able to independently verify it, accurate.

However, readers should distinguish between the investigation's technical findings (which are largely sound) and its geopolitical framing (which involves substantial speculation and appears to reflect political bias in its selective application of suspicion). The technical vulnerability is real and warrants attention; the specific adversary attribution is speculative and not uniquely applicable to Telegram.

Users, security professionals, and organizations making decisions about Telegram adoption should be aware of the `auth_key_id` exposure and its implications for device tracking by network adversaries. This awareness should inform threat modeling, particularly for users facing sophisticated adversaries or operating in high-risk environments where network surveillance is a realistic concern. However, this threat is not unique to Telegram, nor is it uniquely associated with Russian infrastructure—any messaging platform relying on infrastructure providers with government relationships faces comparable theoretical risks, regardless of which government's infrastructure is involved.

5

Privacy Questions & Answers

Legal & Ethical Disclaimer

This report describes technical feasibility only. Any references to "network observers", "visibility to networks", or "logging" refer to technical potential under specific conditions and do not imply lawful or routine practice. The interception, capture, or retention of communications or transport-level metadata (including headers such as `auth_key_id`) without proper legal authorisation — for example a court order or other statutory basis — is unlawful in many jurisdictions. This report does not encourage or endorse any unlawful interception or monitoring; it is intended purely to assess architectural privacy risks and to recommend mitigations.

The technical assessment presented in Chapter 3 and the critical evaluation in Chapter 4 establish that Telegram's MTProto protocol exposes persistent device identifiers—`auth_key_id` values—in cleartext to any network observer positioned between client and server. Our empirical testing confirmed that these 64-bit identifiers remain constant across application restarts, network changes, and extended time periods, contradicting claims of frequent rotation. Furthermore, we demonstrated that both Telegram for Android and Telegram Desktop transmit these identifiers over unencrypted TCP connections, making them trivially observable through standard packet capture techniques.

These technical findings raise fundamental questions about the privacy implications of `auth_key_id` exposure in real-worldf deployment scenarios. While the technical vulnerability is clear—persistent identifiers transmitted without transport-layer encryption—the practical impact depends on numerous factors including the surveillance capabilities of potential adversaries, the legal and technical environments in which Telegram operates, and the feasibility of correlation attacks at Telegram's massive global scale.

This chapter addresses key privacy questions that arise from our technical findings, examining how `auth_key_id` exposure translates into concrete privacy risks across different threat models and deployment contexts. Rather than treating privacy impact as uniform across all scenarios, we analyze how the significance of this vulnerability varies based on:

- The network infrastructure layer at which observation occurs
- The scale and coordination required for global correlation
- The existing surveillance capabilities in different jurisdictions
- The availability of identity anchors and auxiliary data
- The practical requirements for implementing tracking systems

Throughout this analysis, we maintain a distinction between what is technically possible given the protocol vulnerability and what is practically feasible given real-world constraints. However, we emphasize that proper security design should not rely on the practical difficulties faced by adversaries but should instead eliminate unnecessary vulnerabilities through appropriate cryptographic protections.

The following sections examine specific privacy questions through both theoretical analysis and concrete case studies, demonstrating how `auth_key_id` exposure enables various forms of surveillance while also acknowledging the operational challenges that may limit certain attack scenarios. Our goal is to provide a nuanced assessment that neither dismisses the vulnerability as purely theoretical nor exaggerates its impact beyond what the technical evidence supports.

5.1 | Infrastructure Layer Separation and Traffic Access

A critical technical question arises regarding the practical feasibility of extracting `auth_key_id` values from network infrastructure that operates primarily at lower layers of the network stack. Specifically, if an infrastructure provider supplies only Layer 1 (physical layer) services such as DWDM optical channels, while the customer (in this case, Telegram) operates its own routers and manages Layer 3 (network layer) traffic, what are the actual capabilities for traffic observation?

From a network engineering perspective, Layer 1 infrastructure consists of the physical transmission medium—optical fibers, wavelengths, and the equipment that multiplexes and demultiplexes optical signals. A provider operating solely at this layer would typically see only modulated light signals carrying encapsulated data, not the structured IP packets containing MTProto traffic. To access Layer 3 traffic from Layer 1 infrastructure would require several additional steps:

1. **Optical tapping:** The optical signal would need to be split or copied, typically using passive optical splitters or active monitoring equipment.
2. **Signal demodulation:** The optical signal would need to be converted back to electrical signals and demodulated to recover the digital bitstream.
3. **Protocol decoding:** The bitstream would need to be parsed through multiple protocol layers—typically Ethernet frames, IP packets, and TCP segments—to reach the application-layer MTProto data.
4. **Traffic filtering and analysis:** The massive volume of data would need to be filtered to identify Telegram traffic specifically, then analyzed to extract `auth_key_id` values.

These requirements suggest that a pure Layer 1 provider would need to deploy substantial additional active equipment—essentially building a parallel monitoring infrastructure—to access customer traffic. This is technically feasible but represents a significant investment in monitoring capability that goes well beyond the normal operations of a Layer 1 service provider.

However, several important caveats moderate this analysis:

First, the distinction between Layer 1 and Layer 3 operations is not always as clear-cut as theoretical models suggest. Many infrastructure providers operate equipment at multiple layers for operational purposes such as performance monitoring, troubleshooting, and quality assurance. Even ostensibly Layer 1 providers often deploy equipment capable of higher-layer inspection for network management purposes.

Second, the technical capability to access traffic at Layer 1 does exist and is commercially available. Optical network taps, lambda probes, and specialized monitoring

equipment designed for carrier networks can extract and analyze traffic from optical channels. While this requires additional equipment beyond basic DWDM infrastructure, such equipment is neither prohibitively expensive nor technically exotic in the context of major telecommunications infrastructure.

Third, the risk model for `auth_key_id` exposure must consider not just the immediate infrastructure provider but the entire network path. Even if one particular Layer 1 provider cannot or does not access the traffic, the same traffic traverses multiple other network segments where observation may be more straightforward. These include:

- Internet exchange points where traffic is handed off between networks
- Transit providers operating at Layer 3
- Last-mile ISPs serving end users
- Enterprise networks and WiFi access points
- Content delivery networks and reverse proxy services

At any of these points, traffic observation requires no special effort beyond standard packet capture capabilities that are routinely available for legitimate network management purposes.

Fourth, the exposure of `auth_key_id` values through unencrypted transport is a fundamental protocol vulnerability that exists regardless of whether any particular actor chooses to exploit it. The security of a communication system should not depend on assumptions about what adversaries will or will not do, but rather on cryptographic protection that makes exploitation technically infeasible.

Finally, it is worth noting that the capability to observe traffic from optical infrastructure, while requiring additional equipment, is well within the reach of state-level adversaries and sophisticated criminal organizations. The same optical tapping techniques are documented in leaked intelligence documents and are known to be employed by signals intelligence agencies worldwide. The submarine cable tapping operations revealed through various intelligence disclosures demonstrate that even undersea optical cables—arguably the most physically protected Layer 1 infrastructure—are subject to interception.

In conclusion, while pure Layer 1 infrastructure providers may not have immediate, trivial access to Layer 3 traffic containing `auth_key_id` values, the technical barriers to such access are surmountable with moderate investment in monitoring equipment. More importantly, focusing solely on Layer 1 providers understates the exposure risk, as the traffic is readily observable at numerous other points in the network path where Layer 3 access is routine. The fundamental issue remains that transmitting persistent identifiers over unencrypted transport protocols creates an unnecessary and avoidable privacy vulnerability, regardless of which specific network actors might choose

to exploit it. **The proper solution is not to debate the likelihood of exploitation at particular network layers, but to eliminate the vulnerability entirely through mandatory use of transport-layer encryption.**

5.2 | Feasibility of Global Correlation at Telegram's Scale

The question of whether global correlation of `auth_key_id` values is feasible at Telegram's massive scale—with approximately one billion users distributed across hundreds of countries and thousands of network providers—represents a crucial consideration in assessing the real-world privacy impact of this protocol vulnerability. This analysis must consider both the technical challenges and operational realities of implementing such surveillance at planetary scale.

5.2.1 | Scale and Complexity Challenges

The sheer scale of Telegram's infrastructure and user base creates significant challenges for any attempt at comprehensive global tracking:

Volume considerations: With one billion users generating multiple connections per day across different sessions and devices, the total volume of `auth_key_id` observations would easily reach tens of billions of records daily. Each user typically maintains multiple active sessions (phone, desktop, tablet, web), multiplying the tracking complexity. The data storage requirements alone—maintaining historical records for pattern analysis and correlation—would require exabyte-scale infrastructure.

Network diversity: Telegram's traffic traverses an extraordinarily diverse set of network paths:

- Thousands of last-mile ISPs serving end users
- Hundreds of mobile network operators across different countries
- Multiple tier-1 transit providers
- Numerous internet exchange points
- Various cloud infrastructure providers hosting Telegram's servers
- Diverse enterprise networks and public WiFi access points

This diversity means that no single network observation point, or even a small collection of points, could capture more than a fraction of global Telegram traffic.

Geographic distribution: Telegram operates globally with data centers and proxy servers distributed across multiple continents. European users alone connect through hundreds of different carriers and network paths, each potentially using different routes to reach Telegram's infrastructure. The multiplicity of European uplinks mentioned in the question—combined with Telegram's use of content delivery networks and regional points of presence—creates a highly distributed traffic pattern that resists centralized observation.

Dynamic routing: Internet routing is inherently dynamic, with traffic paths changing based on network conditions, peering agreements, and routing policies. The same user connecting to the same Telegram server might traverse different network paths at different times of day, making consistent observation even more challenging.

5.2.2 | Operational and Coordination Barriers

Beyond the technical challenges, implementing global `auth_key_id` correlation faces substantial operational barriers:

Multi-jurisdictional complexity: Coordinating surveillance across multiple legal jurisdictions, each with different privacy laws, data protection regulations, and intelligence-sharing agreements, presents formidable legal and diplomatic challenges. The European Union's GDPR, for instance, creates significant legal barriers to the mass collection and sharing of user tracking data, even when such data is technically available.

Carrier cooperation requirements: Effective global correlation would require active cooperation from hundreds or thousands of network operators, including:

- Installation and maintenance of traffic monitoring equipment
- Real-time or near-real-time data extraction and processing
- Secure data transmission to centralized correlation systems
- Ongoing operational costs and personnel requirements

Many carriers, particularly in privacy-conscious jurisdictions, would likely resist such cooperation absent compelling legal requirements.

Data fusion complexity: Even if data collection were possible at scale, fusing observations from thousands of collection points into a coherent tracking system presents enormous technical challenges. Different carriers use different equipment, logging formats, timestamp synchronization methods, and data schemas. Normalizing and correlating this heterogeneous data in real-time or near-real-time would require sophisticated data processing infrastructure and algorithms.

Identifier rotation impact: If `auth_key_id` values do indeed rotate periodically (though our analysis suggests current rotation is infrequent), this would significantly complicate correlation efforts. Tracking systems would need to:

- Detect rotation events across multiple observation points
- Correlate old and new identifiers to maintain tracking continuity
- Handle edge cases where rotation is observed at some points but not others

- Maintain probabilistic association models when deterministic correlation fails

The complexity increases exponentially with the frequency of rotation and the number of active sessions per user.

5.2.3 | Realistic Threat Scenarios

While comprehensive global correlation may be implausible, this should not diminish concerns about `auth_key_id` exposure. More realistic and equally concerning scenarios include:

Targeted surveillance: Rather than attempting global correlation, adversaries can focus on specific high-value targets. Once a target is identified through any means, their `auth_key_id` enables persistent tracking across any network infrastructure the adversary can access. This targeted approach requires far fewer resources than mass surveillance while still representing a serious privacy violation.

Regional or national-scale tracking: While global correlation may be implausible, national-scale tracking within countries with centralized internet infrastructure is entirely feasible. Countries with state-controlled telecommunications or limited international gateways can more easily implement comprehensive `auth_key_id` monitoring for their populations.

Retrospective analysis: Even without real-time global correlation, the exposure of `auth_key_id` values enables powerful retrospective analysis. Network logs retained for legitimate purposes (troubleshooting, billing, security) can be searched retroactively once a user's `auth_key_id` is known, potentially reconstructing years of communication patterns and relationships.

Federated tracking networks: Instead of a single global correlation system, multiple actors (intelligence agencies, criminal organizations, commercial data brokers) might operate independent tracking systems that occasionally share data. This federated model, while less comprehensive than global correlation, could still achieve significant surveillance coverage for users of interest.

Opportunistic collection: Even without systematic correlation, the widespread availability of `auth_key_id` values in network traffic creates opportunities for opportunistic collection by various actors—from curious system administrators to malicious insiders to criminal organizations operating compromised network infrastructure.

5.2.4 | Assessment and Implications

While truly global, real-time correlation of all Telegram users' `auth_key_id` values is technically and operationally implausible given current realities, this assessment must be carefully qualified:

First, the implausibility of global correlation does not negate the privacy vulnerability. The exposure of persistent identifiers in cleartext remains a fundamental protocol weakness that enables various forms of targeted and opportunistic surveillance, even if comprehensive global tracking is unrealistic.

Second, technological advancement tends to make previously implausible surveillance capabilities increasingly feasible over time. The computational and storage capabilities that would make global correlation implausible today may become readily available in the future, while `auth_key_id` values collected today remain useful for retroactive analysis.

Third, the focus on global correlation may distract from more achievable but still harmful surveillance scenarios. Regional tracking, targeted surveillance, and retrospective analysis all remain feasible and concerning, particularly for vulnerable populations such as political dissidents, journalists, and human rights activists.

Fourth, the argument that global correlation is implausible essentially relies on the failure of adversaries to cooperate effectively—a dangerous assumption in security engineering. Proper security design should not depend on the incompetence or lack of coordination among adversaries.

Finally, even if we accept that global correlation is currently implausible, this represents a risk assessment rather than a security guarantee. The presence of unencrypted `auth_key_id` values in network traffic creates a latent capability that could be exploited should circumstances change—whether through technological advancement, political shifts, or increased adversary motivation.

In conclusion, while the scale and complexity of Telegram's global operations do indeed make comprehensive, real-time correlation of all users' `auth_key_id` values technically and operationally implausible with current technology and organizational structures, this should not be interpreted as validating the protocol design. The exposure of persistent identifiers over unencrypted channels remains a serious privacy vulnerability that enables various realistic attack scenarios. The proper response is not to debate the feasibility of the most extreme surveillance scenarios, but to eliminate the underlying vulnerability through mandatory transport-layer encryption. The cost of implementing proper encryption is negligible compared to the privacy risks created by the current design, regardless of whether those risks manifest as global correlation or more targeted forms of surveillance.

5.3 | Evaluating auth_key_id in Varied Surveillance Environments

A nuanced analysis of auth_key_id exposure must consider how its intelligence value varies across different surveillance environments. Two contrasting scenarios illuminate this variation: jurisdictions with pervasive network monitoring and comprehensive IP attribution (such as Russia), versus privacy-protective jurisdictions with legal restrictions on mass surveillance (such as the European Union). This analysis reveals that auth_key_id provides distinct and significant value in both environments, though for different reasons.

5.3.1 | High-Surveillance Environments: Beyond IP Attribution

In environments with pervasive network monitoring, where state agencies already maintain comprehensive databases linking IP addresses to real-world identities, one might reasonably question what additional value auth_key_id collection provides. If authorities can already attribute any IP address to a specific subscriber through mandatory ISP logging, why would tracking an additional device identifier matter?

This perspective significantly underestimates the unique intelligence value that auth_key_id provides, even in environments with complete IP attribution:

Device-level granularity: While IP addresses identify network connections, they do not reliably identify individual devices. In typical scenarios:

- Multiple household members share a single residential IP address
- Corporate and institutional networks use NAT, placing hundreds or thousands of devices behind single IP addresses
- Mobile carriers use carrier-grade NAT (CGNAT), where thousands of users may share IP addresses
- Public WiFi networks serve numerous anonymous users through single IP addresses
- VPN and proxy services deliberately obscure the relationship between users and IP addresses

The auth_key_id cuts through this ambiguity by providing a persistent identifier for each specific Telegram installation, enabling authorities to distinguish between different users of shared network infrastructure and track specific devices even as they move between networks.

Cross-network tracking continuity: Users constantly move between different networks—home WiFi, cellular data, office networks, public hotspots—receiving different IP addresses with each transition. While IP-based tracking loses continuity at

each network change, `auth_key_id` provides uninterrupted tracking across all these transitions. This enables authorities to:

- Track user movements through their network connection patterns
- Identify when users attempt to evade surveillance by changing networks
- Correlate activities across different network contexts
- Build comprehensive communication profiles spanning all network environments

Historical correlation and retroactive surveillance: The persistence of `auth_key_id` enables powerful retroactive analysis capabilities that IP addresses alone cannot provide. When a person of interest is identified, authorities can search historical network logs for their `auth_key_id` to:

- Reconstruct communication patterns from before they became surveillance targets
- Identify historical network locations and movement patterns
- Discover previously unknown associates through temporal correlation
- Build evidence of long-term behavioral patterns

This retroactive capability is particularly valuable in jurisdictions where authorities maintain extensive historical network logs but may not actively monitor all citizens in real-time.

Multi-device tracking and session correlation: Modern users typically operate multiple Telegram sessions simultaneously—phone, tablet, desktop, web browser—each with its own `auth_key_id`. While IP addresses might temporarily coincide when devices share a network, the distinct `auth_key_id` values enable authorities to:

- Map the complete device ecosystem of surveillance targets
- Track which specific device was used for particular communications
- Identify when users acquire new devices or create new sessions
- Detect attempts to compartmentalize communications across different devices

Circumvention detection and resilience: Even in high-surveillance environments, users attempt to evade monitoring through various means—VPNs, Tor, proxy services, SIM card swapping, or using others' devices. The `auth_key_id` provides a tracking anchor that persists through many of these evasion attempts:

- VPN usage changes IP addresses but not `auth_key_id` values
- Proxy services obscure network paths but preserve application-layer identifiers
- SIM swapping changes phone numbers but maintains device continuity
- Borrowed WiFi networks show different IPs but same device identifiers

This resilience to common circumvention techniques makes `auth_key_id` particularly valuable for maintaining surveillance continuity on high-value targets who actively attempt counter-surveillance.

Social graph construction and relationship mapping: The combination of IP attribution and `auth_key_id` enables sophisticated social network analysis that neither identifier alone could achieve. Authorities can:

- Identify when different `auth_key_id` values appear on the same IP address, suggesting physical proximity or relationship
- Track group gatherings through convergence of multiple known `auth_key_id` values
- Discover hidden relationships through pattern analysis of identifier co-occurrence
- Map organizational structures through device interaction patterns

5.3.2 | Privacy-Protective Environments: Enabling Targeted Surveillance

In jurisdictions with strong privacy protections, where mass IP-level attribution is legally restricted, the question becomes whether `auth_key_id` data alone can meaningfully compromise user privacy without auxiliary datasets linking identifiers to real-world identities.

Initial anchor establishment: While mass surveillance may be restricted, targeted surveillance of specific individuals typically remains legal with appropriate authorization. Once authorities establish even a single correlation between an `auth_key_id` and a real-world identity—through targeted investigation, legal compulsion, physical surveillance, or technical means—that identifier becomes permanently compromised. The persistence of `auth_key_id` means this single anchor point enables ongoing tracking that would be impossible with dynamic identifiers.

Opportunistic correlation: Even without mass surveillance programs, numerous opportunities exist for correlating `auth_key_id` values with identities:

- Commercial WiFi hotspots that require registration or payment

- Corporate networks where device registration is mandatory
- Hotel networks requiring room number authentication
- Airport and transportation WiFi with identity verification
- Healthcare facilities with patient network access
- Educational institutions with student authentication systems

Each of these touchpoints creates potential correlation opportunities that transform anonymous `auth_key_id` values into attributed identities.

Behavioral fingerprinting and pattern analysis: Even without direct identity correlation, `auth_key_id` enables behavioral analysis that can lead to deanonymization:

- Communication timing patterns (when someone is active) can suggest timezone and schedule
- Network location patterns (which cell towers or WiFi networks are used) reveal movement habits
- Social interaction patterns (who communicates with whom) expose relationship networks
- Content correlation (matching writing style or interests across platforms) enables cross-platform tracking

Research has repeatedly demonstrated that behavioral patterns alone can deanonymize users with high accuracy, even without initial identity anchors.

Gradual intelligence accumulation: Intelligence gathering in privacy-protective environments often operates through gradual accumulation rather than mass collection. `auth_key_id` values enable this approach by:

- Allowing long-term tracking of interesting but unidentified devices
- Building behavioral profiles over time until identification becomes possible
- Maintaining surveillance continuity across different legal authorities and timeframes
- Creating persistent records that remain valuable when circumstances change

Cross-border intelligence sharing: While the EU has strong privacy protections, intelligence sharing agreements and mutual legal assistance treaties create pathways for

information exchange. An `auth_key_id` identified in a less privacy-protective jurisdiction can be tracked when the device enters the EU, effectively importing surveillance capabilities across borders.

Non-state actors and criminal exploitation: Privacy laws restrict government surveillance but may not prevent collection by other actors:

- Criminal organizations operating compromised infrastructure
- Foreign intelligence services not bound by local laws
- Commercial data brokers operating in legal gray areas
- Malicious insiders at ISPs or infrastructure providers
- Hackers who compromise network equipment or databases

For these actors, `auth_key_id` values provide tracking capabilities regardless of legal restrictions on mass surveillance.

Retroactive deanonymization: Perhaps most concerning is the retroactive deanonymization risk. `auth_key_id` values collected today under privacy-protective regimes remain vulnerable to future deanonymization through:

- Changes in privacy laws or surveillance authorities
- Data breaches exposing identity correlation databases
- Advances in deanonymization techniques and behavioral analysis
- Accumulation of additional correlation data over time
- Political shifts that alter surveillance practices

5.3.3 | Comparative Assessment and Implications

The analysis reveals that `auth_key_id` exposure provides significant surveillance value in both high-surveillance and privacy-protective environments, though through different mechanisms:

In **high-surveillance environments**, `auth_key_id` complements existing IP attribution by providing:

- Device-specific tracking that IP addresses cannot deliver
- Cross-network tracking continuity
- Resilience against common circumvention techniques

- Enhanced capability for retroactive surveillance
- Sophisticated social network analysis

In **privacy-protective environments**, `auth_key_id` enables:

- Targeted tracking following initial identification
- Opportunistic correlation through various touchpoints
- Behavioral analysis leading to deanonymization
- Gradual intelligence accumulation over time
- Exploitation by non-state actors outside legal frameworks

The postulate that `auth_key_id` adds "little practical advantage" in high-surveillance environments or is "insufficient for actionable deanonymization" in privacy-protective environments fundamentally misunderstands modern surveillance capabilities and practices. In reality:

First, the value proposition differs by environment but remains significant in both. High-surveillance environments gain device-level precision and circumvention resistance, while privacy-protective environments gain targeted tracking capabilities that would otherwise be unavailable.

Second, the persistence of these identifiers—even if rotation occurs every 24 hours (which our analysis suggests is not the current implementation)—provides sufficient continuity for most surveillance purposes. Daily rotation would complicate but not prevent tracking, particularly when combined with behavioral analysis and pattern recognition.

Third, the binary framing of "complete IP attribution" versus "no attribution" oversimplifies real-world surveillance landscapes. Most environments fall somewhere between these extremes, with varying degrees of logging, retention, and access controls. `auth_key_id` exposure creates vulnerabilities across this entire spectrum.

Fourth, focusing solely on government surveillance ignores the broader threat landscape. Criminal organizations, foreign intelligence services, and commercial entities all benefit from `auth_key_id` exposure, regardless of local legal frameworks.

Finally, the argument implicitly accepts unnecessary privacy risks. Even if `auth_key_id` provided only marginal additional surveillance capability (which this analysis refutes), the proper question is why any unnecessary identifying information should be exposed when transport encryption could eliminate this exposure at negligible cost.

In conclusion, `auth_key_id` exposure provides significant surveillance value in both high-surveillance and privacy-protective environments, enabling capabilities that

would not exist with proper transport encryption. The attempt to minimize this vulnerability by arguing it provides little additional value in either environment type reflects a misunderstanding of modern surveillance practices and the diverse ways persistent identifiers can be exploited. The appropriate response is not to debate marginal surveillance values but to implement proper encryption that eliminates these unnecessary exposures entirely.

5.4 | Illustrative Case Study

To illustrate the real-world implications of `auth_key_id` exposure, we present a hypothetical but technically accurate scenario demonstrating how these persistent identifiers enable comprehensive tracking and eventual deanonymization of a high-risk user.

5.4.1 | Hypothetical Scenario: Journalist Deanonymization Timeline

This illustrative example follows an investigative journalist using Telegram to communicate with sources about government corruption. While hypothetical, every technical detail described is based on actual capabilities of existing network infrastructure and commercially available monitoring systems. The scenario demonstrates how `auth_key_id` exposure enables tracking across multiple networks and eventual identity attribution through opportunistic correlation.

Day 1 - Initial Collection (08:00): The journalist connects to their home WiFi network and opens Telegram Desktop on their laptop. Their ISP, operating under data retention regulations, logs:

- Timestamp: 2024-01-15 08:00:00
- Source IP: 192.168.1.42 (NAT to public IP 203.0.113.15)
- Destination: Telegram server 149.154.167.51:443
- Extracted `auth_key_id`: `fa:ac:d7:21:91:22:db:87`
- Session duration: 45 minutes

The ISP does not know the journalist's identity at this point—the internet subscription is in their roommate's name. The `auth_key_id` is logged but remains unattributed.

Day 3 - Mobile Network Correlation (14:30): The journalist travels to meet a source, using Telegram for Android on their phone via cellular data. The mobile carrier's deep packet inspection system, deployed for "network optimization," extracts and logs:

- Timestamp: 2024-01-17 14:30:00
- Source: Mobile subscriber IMSI-123456789012345
- Cell tower location: 52.5200°N, 13.4050°E (Berlin city center)
- Extracted `auth_key_id`: `09:da:84:02:bb:23:81:11` (phone app)
- Concurrent `auth_key_id`: `fa:ac:d7:21:91:22:db:87` (desktop app in background sync)

The carrier knows the phone subscriber's identity but has no immediate reason to flag this traffic. However, the correlation between phone and desktop `auth_key_id` values is now recorded in their logs.

Day 7 - Anchor Point Establishment (10:15): The journalist attends a press conference at a government building. To access the guest WiFi, they must register with their press credentials:

- Registration: Name, outlet, press ID number
- Device MAC address: aa:bb:cc:dd:ee:ff
- Assigned IP: 10.50.1.234
- `auth_key_id` observed in traffic: `fa:ac:d7:21:91:22:db:87`

This creates the critical anchor point: the `auth_key_id` `fa:ac:d7:21:91:22:db:87` is now definitively linked to the journalist's real identity through the press credential registration.

Day 10 - Airport Surveillance (16:45): Traveling to meet another source, the journalist uses airport WiFi:

- Airport security logs show entry at 15:30
- WiFi captive portal logs email for "free 30 minutes"
- `auth_key_id` `fa:ac:d7:21:91:22:db:87` observed from 16:45-17:30
- Flight manifest shows journalist on 18:00 flight to Vienna

The airport's integration of physical security, network access, and passenger data creates additional identity confirmation.

Day 14 - VPN Circumvention Attempt (20:00): Aware of potential surveillance, the journalist activates a commercial VPN:

- Home ISP logs VPN connection to 198.51.100.50
- VPN provider's netflow logs (retained despite "no-logs" policy) show:
- Incoming connection from 203.0.113.15
- Outgoing Telegram traffic to 149.154.167.51
- `auth_key_id` `fa:ac:d7:21:91:22:db:87` clearly visible in unencrypted MTProto

The VPN successfully masks the IP address from Telegram's perspective but does not protect the `auth_key_id` from observation by the VPN provider or any infrastructure between the VPN exit and Telegram servers.

Day 20 - Retroactive Analysis: Following publication of the corruption investigation, authorities subpoena network logs from multiple providers. Using the anchor point from Day 7, they search for `auth_key_id` `fa:ac:d7:21:91:22:db:87` across all available logs:

- Home internet usage patterns reveal working hours and habits
- Mobile network logs expose movement patterns and meeting locations
- Correlation with source devices (other `auth_key_id` values appearing at same times/locations)
- Communication timing analysis suggests specific contacts
- VPN usage patterns indicate awareness of surveillance

Infrastructure Required: This hypothetical deanonymization required only:

- Standard ISP logging infrastructure (already deployed for troubleshooting/-billing)
- Mobile carrier DPI systems (commercially available from vendors like Sandvine, Procera)
- Captive portal authentication logs (standard in enterprise/public WiFi)
- Legal authority to compel log disclosure (standard subpoena/warrant)
- Simple database queries to correlate `auth_key_id` across datasets

No sophisticated nation-state capabilities or mass surveillance infrastructure was necessary—only the correlation of logs that providers already maintain for operational purposes.

5.4.2 | Key Implications of This Scenario

While hypothetical, this scenario illustrates several critical points about `auth_key_id` exposure:

Single Anchor Point Vulnerability: A single correlation between an `auth_key_id` and a real-world identity (the press conference WiFi registration) compromised all past and future communications using that identifier. Once established, this anchor point enabled retroactive analysis of all logged traffic.

Standard Infrastructure Sufficiency: The tracking and deanonymization did not require specialized surveillance equipment or mass monitoring capabilities. Standard logging systems already deployed by ISPs, mobile carriers, and WiFi operators provided all necessary data.

VPN Ineffectiveness: The journalist's attempt to use a VPN for anonymity failed because `auth_key_id` values remain visible in the unencrypted MTProto protocol. This demonstrates that common privacy tools cannot protect against this vulnerability.

Cross-Network Correlation: The persistent `auth_key_id` enabled tracking across completely different network types (home ISP, mobile carrier, public WiFi, VPN), creating a comprehensive picture of the journalist's movements and communication patterns.

Retroactive Intelligence Value: Logs collected without specific surveillance intent became valuable intelligence after the journalist was identified. This illustrates how `auth_key_id` exposure creates latent surveillance capabilities that can be activated when needed.

This hypothetical scenario, while focused on a journalist, applies equally to other high-risk users including political activists, human rights defenders, and anyone whose communications might attract scrutiny from powerful actors. The vulnerability is not theoretical—it represents a practical risk that could be exploited today using existing infrastructure and commercially available tools.

Most critically, this entire tracking and deanonymization capability would be eliminated if Telegram implemented proper transport layer encryption. The persistence of this vulnerability despite the availability of straightforward technical solutions raises serious questions about the platform's commitment to user privacy.

6

Architectural Responsibility and Liability

A critical aspect that emerges from our technical analysis concerns the assignment of responsibility and liability for the `auth_key_id` exposure vulnerability. The OCCRP investigation focuses significant attention on Telegram's infrastructure providers, particularly those with potential connections to surveillance agencies. However, this framing fundamentally misallocates responsibility for what is, at its core, an architectural failure on Telegram's part.

6.1 | The Root Cause: Telegram's Design Choice

The exposure of `auth_key_id` values to network observers is not an inevitable consequence of operating a messaging platform. It is not a necessary trade-off for performance, scalability, or functionality. It is not imposed by infrastructure limitations or network requirements. Rather, it is the direct and predictable result of Telegram's conscious decision not to implement transport-layer encryption using industry-standard protocols like TLS.

Every major messaging platform—Signal, WhatsApp, iMessage, and others—demonstrates that secure messaging can be delivered while encrypting the transport layer. These platforms wrap their application protocols in TLS, ensuring that any application-layer identifiers, session tokens, or metadata remain opaque to network intermediaries. The computational overhead of TLS in modern implementations is negligible, particularly compared to the cryptographic operations already performed by MTProto itself.

Telegram's choice to transmit MTProto over unencrypted TCP connections—whether on ports 80, 443, or 5222—represents a deliberate architectural decision that prioritizes other considerations over the fundamental privacy principle of minimizing metadata exposure. This decision cannot be justified on technical grounds when proven alternatives exist and are widely deployed.

6.2 | Misplaced Liability

By exposing `auth_key_id` values in cleartext, Telegram effectively transforms every network intermediary into a potential surveillance point. This includes:

- Infrastructure providers operating physical and optical networks
- Datacenter operators providing colocation and connectivity
- Internet Service Providers serving end users
- Transit providers carrying traffic between networks
- Internet exchange points facilitating peering
- Corporate and institutional network administrators
- Public WiFi operators

None of these entities chose to have visibility into Telegram users' persistent device identifiers. They did not request this capability, design systems to exploit it, or necessarily even realize they possess it. They are simply transporting network packets according to standard routing protocols, fulfilling their fundamental operational purpose.

Yet Telegram's architectural choices force these infrastructure partners to handle sensitive metadata that should never be their concern. An optical network provider delivering Layer 1 connectivity should not have to worry about whether their equipment might observe user tracking identifiers. A datacenter operator providing rack space and power should not bear responsibility for the privacy implications of unencrypted application protocols. An ISP routing packets should not be placed in the position of possessing persistent device identifiers simply because one application chose not to implement transport encryption.

6.3 | The Liability Transfer

This architectural decision represents a transfer of liability from Telegram to its entire infrastructure ecosystem. Consider the practical implications:

- **Legal exposure:** Infrastructure providers may face legal obligations to log, retain, or disclose the metadata they can observe, even if they would prefer not to have such visibility.
- **Reputational risk:** Partners are implicated in privacy controversies not due to their own actions but because Telegram's protocol makes observation technically possible.
- **Operational burden:** Providers must implement policies and procedures for handling sensitive data they should never see in the first place.
- **Ethical dilemmas:** Network operators are forced to make decisions about whether to examine, log, or act upon visible metadata that proper encryption would have rendered inaccessible.

6.4 | False Equivalence in Attribution

The OCCRP investigation devotes substantial effort to examining the business relationships and potential intelligence connections of Telegram's infrastructure providers, particularly those with ties to Russia. While such scrutiny of infrastructure relationships has investigative merit, it creates a false equivalence that obscures the fundamental issue: the vulnerability exists because of Telegram's protocol design, not because of who operates the infrastructure.

If Telegram implemented proper transport-layer encryption, the identity and trustworthiness of infrastructure providers would be largely irrelevant to user privacy at the network layer. The packets would be opaque bytes requiring no trust in intermediate handlers. The current scrutiny of infrastructure providers is only necessary because Telegram's architecture makes such trust essential.

Furthermore, the selective focus on Russian-connected infrastructure providers implies that Western infrastructure would be inherently more trustworthy. This is a dangerous assumption that ignores the documented history of surveillance programs operated by Western intelligence agencies, the existence of National Security Letters and similar legal instruments, and the commercial surveillance ecosystem that monetizes user data. The problem is not which specific entities operate the infrastructure, but that the infrastructure has unnecessary visibility in the first place.

6.5 | Industry Standards and Best Practices

The security community has long established clear best practices for protecting user privacy in communication systems:

- **Defense in depth:** Multiple layers of protection ensure that the failure of one layer does not compromise the entire system.
- **Least privilege:** Systems and actors should have access only to the minimum information necessary for their function.
- **End-to-end principle:** Security properties should be enforced as close to the endpoints as possible, not relying on intermediate infrastructure.
- **Cryptographic protection:** Sensitive data should be encrypted whenever it leaves the direct control of trusted systems.

Telegram's failure to implement transport-layer encryption violates all of these principles. It creates a single point of failure (network observation), grants excessive privilege to infrastructure (visibility of tracking identifiers), relies on infrastructure trustworthiness (rather than cryptographic protection), and transmits sensitive identifiers in cleartext across untrusted networks.

6.6 | The Simple Solution

The technical solution to eliminate `auth_key_id` exposure is straightforward and well-understood: mandate the use of TLS for all MTProto connections. This would:

- Render `auth_key_id` values invisible to passive network observers
- Remove the ability for infrastructure providers to track devices
- Eliminate the liability currently imposed on network intermediaries
- Align Telegram with industry-standard security practices
- Require minimal changes to the existing protocol architecture

The implementation complexity is modest—TLS libraries are mature, well-tested, and available for all platforms where Telegram operates. The performance impact is negligible, particularly given that Telegram already performs cryptographic operations for MTProto. The only barrier to implementation appears to be Telegram's organizational will to prioritize user privacy over whatever considerations currently prevent this obvious improvement.

The debate should not focus on which infrastructure providers Telegram uses, what their relationships might be with intelligence agencies, or how likely they are to exploit the visible metadata. These are secondary concerns that arise only because of Telegram's fundamental architectural failure. The primary issue is that Telegram has chosen not to implement basic transport-layer encryption, thereby exposing persistent device identifiers to every network intermediary and creating unnecessary privacy risks for its users.

This is not a complex technical challenge requiring breakthrough innovation. It is not a trade-off necessitated by competing requirements. It is a straightforward engineering decision where Telegram has chosen the less secure option despite clear alternatives. Until Telegram implements mandatory transport-layer encryption, it bears full responsibility for the privacy implications of `auth_key_id` exposure and for the liability it imposes on its entire infrastructure ecosystem. The solution is not to scrutinize infrastructure providers more carefully, but to eliminate their ability to observe sensitive metadata through proper cryptographic protection.

7

Conclusions

This report examined the privacy implications of Telegram's MTProto protocol implementation, with particular focus on the exposure of authorization key identifiers (`auth_key_id`) through unencrypted transport layers.

Our technical analysis, which included independent testing of Telegram for Android and Telegram for Desktop (macOS), confirmed several critical findings:

1. **Unencrypted transport exposure:** Both Telegram for Android and Telegram for Desktop transmit MTProto over unencrypted TCP connections rather than HTTPS. While Telegram Desktop uses port 443 (commonly associated with encrypted HTTPS traffic), the actual traffic is plain TCP without TLS encryption. This design choice exposes the entire MTProto external header, including the `auth_key_id`, to passive network observation.

2. **Persistent device identifiers:** The `auth_key_id` functions as a persistent, device-specific identifier that remains constant across application restarts, IP address changes, network switches, and connections to different Telegram servers. Even when PFS is enabled and temporary authorization keys are used, the temporary `auth_key_id` values persist for their entire validity period (typically 24 hours) and transitions between temporary keys are readily observable and linkable.

3. **Trivial obfuscation:** MTProto applies an obfuscation layer to its payloads, but this obfuscation is cryptographically trivial and explicitly designed only to prevent naive protocol detection rather than provide security. The obfuscation can be reversed using publicly documented techniques, making `auth_key_id` extraction straightforward for any network observer.

4. **Broad adversary visibility:** The use of unencrypted transport means that `auth_key_id` values are visible to an extremely broad range of potential adversaries, including ISPs, network administrators, government surveillance programs, malicious hotspot operators, and any party capable of passive traffic monitoring. No active attack, certificate compromise, or protocol manipulation is required—simple packet capture and trivial deobfuscation suffice.

5. **Cross-platform consistency:** The `auth_key_id` exposure affects both mobile and desktop clients, suggesting this is a systematic architectural decision rather than an isolated implementation oversight. Users cannot protect themselves from `auth_key_id`-based tracking by switching between client platforms.

6. **Ineffectiveness of PFS:** MTProto's PFS implementation does not mitigate `auth_key_id`-based tracking. While PFS provides important protections against post-compromise decryption of message contents, it does nothing to prevent device tracking through observable identifier values in unencrypted transport-layer metadata.

7. **Secret Chats equally affected:** End-to-end encryption in Secret Chats protects message content from observation by Telegram's servers and network intermediaries, but does not prevent device tracking through `auth_key_id` observation. The tracking vulnerability operates at the MTProto transport layer, which underlies the Secret Chat encryption layer.

7.1 | Evaluation of Public Claims

We evaluated five principal claims from the OCCRP investigation [1] regarding Telegram's privacy properties:

- **Claim 1 (Unencrypted `auth_key_id` enables user tracking):** Fully supported by our independent technical analysis. The claim is accurate without overstatement.
- **Claim 2 (Network access enables metadata surveillance):** Substantially accurate regarding the technical capabilities of traffic analysis and metadata extraction. However, certain formulations overstate the degree to which communication graphs can be reconstructed through network observation alone in a centralized server architecture, and the claim presents metadata surveillance capabilities as though they were unique to Telegram when they are in fact common to virtually all internet-based messaging platforms.
- **Claim 3 (End-to-end encryption does not prevent tracking):** Fully supported by our analysis. The claim correctly identifies that Secret Chats' end-to-end encryption protects message content but not device-level tracking through `auth_key_id` observation at the transport layer.
- **Claim 4 (Infrastructure control enables surveillance):** The technical core is accurate—infrastructure-level access combined with `auth_key_id` visibility could enable extensive device tracking. However, the claim blends sound technical analysis with geopolitical threat modeling and specific adversary attribution based on circumstantial evidence. The investigation's methodology involves guilt-by-association reasoning through multiple degrees of separation that could equally be applied to Western messaging platforms and their infrastructure providers, revealing apparent political bias in the selective application of suspicion.
- **Claim 5 (Router control enables tracking):** Fully supported by our empirical findings. The claim is technically accurate and appropriately scoped.

Overall, the technical claims regarding `auth_key_id` exposure and tracking capability are predominantly accurate and independently reproducible. Where claims extend into geopolitical attribution or subjective risk characterization, we note these elements as beyond the scope of technical validation.

7.2 | Privacy Impact Analysis

Our expanded analysis in Chapters 5 and 6 revealed nuanced but significant privacy implications across varied deployment contexts:

Infrastructure layer considerations: While Layer 1 optical infrastructure providers face technical barriers to accessing Layer 3 traffic containing `auth_key_id` values, these barriers are surmountable with moderate investment in monitoring equipment. More critically, the exposure occurs at numerous other points in the network path where Layer 3 access is routine, including ISPs, exchange points, and enterprise networks.

Scale and correlation challenges: Global correlation of all Telegram users' `auth_key_id` values faces substantial technical and operational challenges given the platform's billion-user scale and geographic distribution. However, this does not diminish the vulnerability's significance for:

- Targeted surveillance of specific individuals
- Regional or national-scale tracking within centralized network environments
- Retrospective analysis once identity anchors are established
- Opportunistic collection by diverse threat actors

Varied surveillance environments: The `auth_key_id` provides distinct value in both high-surveillance and privacy-protective jurisdictions:

- In high-surveillance environments, it enables device-level tracking granularity beyond what IP attribution alone provides
- In privacy-protective environments, it creates targeted tracking capabilities that would otherwise be legally restricted
- In both contexts, the persistence of identifiers enables powerful retroactive analysis

Practical exploitation scenarios: Our hypothetical case study demonstrated how standard network infrastructure and routine logging practices enable comprehensive tracking and eventual deanonymization through opportunistic correlation. No sophisticated nation-state capabilities are required—only the correlation of logs that providers already maintain for operational purposes.

7.3 | Architectural Responsibility

Chapter 6 established that the `auth_key_id` exposure represents a fundamental architectural failure on Telegram's part, not a consequence of infrastructure choices:

Root cause analysis: The vulnerability stems directly from Telegram's decision not to implement transport-layer encryption, despite:

- Industry-standard practices mandating TLS for sensitive communications
- Negligible performance overhead of modern TLS implementations
- Successful deployment of transport encryption by all major competing platforms
- Clear security principles requiring cryptographic protection of sensitive metadata

Misplaced liability: Telegram's architecture transfers privacy liability to its entire infrastructure ecosystem, forcing network intermediaries to handle sensitive metadata they should never see. This creates legal exposure, reputational risks, and ethical dilemmas for providers who have no control over the protocol design.

False equivalencies: While infrastructure provider relationships merit scrutiny, focusing on specific providers' trustworthiness obscures the fundamental issue: proper transport encryption would render infrastructure trust irrelevant to user privacy at the network layer.

7.4 | Implications and Recommendations

The comprehensive analysis presented in this report yields clear implications for different stakeholders:

For users: Telegram users should understand that:

- Device identifiers are visible to any network observer, enabling persistent tracking
- This vulnerability cannot be mitigated through user configuration or privacy features
- VPNs provide limited protection due to the unencrypted protocol design
- High-risk users should consider this exposure in their threat models

For security professionals: Organizations must recognize that:

- Standard encrypted messaging threat models do not apply to Telegram
- The platform creates unique risks for users under network surveillance
- Alternative platforms with proper transport encryption should be preferred for sensitive communications
- Risk assessments must account for both targeted and opportunistic tracking scenarios

For Telegram: The remediation path is straightforward:

- Mandate TLS encryption for all MTProto connections
- Remove support for unencrypted transport options
- Coordinate updates across all client platforms
- Accept the modest computational costs as necessary for user privacy

Contextual perspective: While the `auth_key_id` exposure represents a serious Telegram-specific vulnerability, readers should recognize that:

- Many metadata surveillance capabilities affect all messaging platforms equally
- The selective focus on certain infrastructure relationships reflects political framing

- The core technical vulnerability remains real regardless of geopolitical narratives
- Proper security design should eliminate unnecessary vulnerabilities regardless of threat likelihood

7.5 | Limitations of This Study

Our analysis focused on Telegram for Android and Telegram for Desktop (macOS). While Telegram Desktop shares a common codebase across Windows, macOS, and Linux platforms, we did not independently verify iOS, Windows, or Linux clients. The findings should be considered highly likely to apply to these platforms given architectural consistency, but definitive confirmation would require platform-specific testing.

Our testing examined standard Telegram usage scenarios over conventional internet connections. We did not comprehensively test all of Telegram's transport options, proxy configurations, or specialized deployment scenarios. Our findings reflect Telegram's default behavior as experienced by typical users.

The geopolitical claims regarding specific adversaries' access to infrastructure and potential surveillance capabilities are beyond the scope of technical analysis and were evaluated only for logical consistency and evidentiary support, not through independent intelligence assessment.

7.6 | Conclusion

Telegram's decision to transmit MTProto over unencrypted transport layers creates a significant and systematic privacy vulnerability that enables device tracking by passive network observers. This architectural choice, which contradicts established security practices and industry standards, exposes persistent `auth_key_id` values to an unnecessarily broad range of potential adversaries.

Our technical analysis confirmed that this vulnerability is real, reproducible, and affects all major Telegram client platforms. The exposure enables various surveillance scenarios ranging from targeted tracking of specific individuals to opportunistic collection by malicious actors. While global-scale correlation faces practical challenges, the vulnerability's impact remains significant across diverse threat models and deployment contexts.

The analysis of varied surveillance environments demonstrated that `auth_key_id` provides intelligence value in both high-surveillance and privacy-protective jurisdictions, though through different mechanisms. The hypothetical case study illustrated how standard infrastructure and routine logging practices enable comprehensive tracking without requiring sophisticated capabilities.

Most critically, this vulnerability exists solely due to Telegram's architectural decisions, not because of infrastructure limitations or technical constraints. The platform's failure to implement mandatory transport-layer encryption represents a transfer of liability to infrastructure providers and an unnecessary exposure of user privacy.

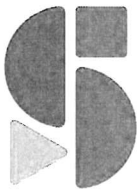
The technical findings of this report are independently reproducible and supported by empirical evidence. While public reporting has sometimes conflated technical analysis with geopolitical speculation, the core vulnerability warrants serious attention regardless of political narratives.

Users of Telegram should incorporate awareness of `auth_key_id` exposure into their threat models, particularly in contexts where network surveillance is a realistic concern. Security professionals should recognize that standard encrypted messaging threat models do not apply to Telegram due to this transport-layer vulnerability.

Telegram should prioritize implementing mandatory TLS encryption for all MTProto connections. This straightforward technical solution would eliminate the vulnerability entirely, align the platform with industry standards, and demonstrate genuine commitment to user privacy. Until such measures are implemented, Telegram cannot credibly claim to prioritize user privacy while maintaining an architecture that unnecessarily exposes persistent tracking identifiers to network observers.

8

About Symbolic Software



Symbolic Software[1], established in Paris, France in 2017, is a software consultancy specializing in applied cryptography and software security. The firm has executed over 300 cryptographic software audits within the European information security sector and has made significant contributions to the field by publishing peer-reviewed cryptographic research software.

Offering wide-ranging expertise in cryptographic software audits, Symbolic Software has audited critical cryptographic components of global platforms, ranging from password managers to cryptocurrencies. The company has developed Verifpal® and Noise Explorer, innovative research software for cryptographic engineering, which have contributed to peer-reviewed scientific publications. Symbolic Software's portfolio is marked by collaboration with leading entities such as Cure53 and the Linux Foundation, and they have successfully audited critical technologies like MetaMask and key COVID-19 contact tracing applications in Europe.

[1] Stay updated on Symbolic Software's latest work by visiting `https://symbolic.sofware`.

Bibliography

[1] Organized Crime and Corruption Reporting Project. *Telegram, the FSB, and the Man in the Middle*. 2025. URL: https://www.occrp.org/en/investigation/telegram-the-fsb-and-the-man-in-the-middle (visited on 10/06/2025).

[2] Telegram FZ-LLC. *MTProto Mobile Protocol*. 2025. URL: https://core.telegram.org/mtproto (visited on 10/07/2025).

[3] Telegram FZ-LLC. *Mobile Protocol: Detailed Description*. 2025. URL: https://core.telegram.org/mtproto/description (visited on 10/07/2025).

[4] Telegram FZ-LLC. *Mobile Protocol: Transports*. 2025. URL: https://core.telegram.org/mtproto/transports (visited on 10/07/2025).

[5] Michał "rysiek" Woźniak. *Telegram is indistinguishable from an FSB honeypot*. 2025. URL: https://rys.io/en/179.html (visited on 10/07/2025).

[6] WhatsApp. *WhatsApp Encryption Overview: Technical white paper*. 2024. (Visited on 10/07/2025).

[7] Moxie Marlinspike. *A letter from Amazon*. 2018. URL: https://signal.org/blog/looking-back-on-the-front/ (visited on 10/07/2025).

[8] Telegram FZ-LLC. *Telegram API: Perfect Forward Secrecy*. 2025. URL: https://core.telegram.org/api/pfs (visited on 10/07/2025).

[9] Telegram FZ-LLC. *Telegram Desktop messaging app*. 2025. URL: https://github.com/telegramdesktop/tdesktop (visited on 10/07/2025).

[10] Frank Konkel. *The Details About the CIA's Deal With Amazon*. 2014. URL: https://www.theatlantic.com/technology/archive/2014/07/the-details-about-the-cias-deal-with-amazon/374632/ (visited on 10/07/2025).

[11] Meta, Inc. *Integrity Reports, First Quarter 2025*. 2025. URL: https://transparency.meta.com/en-gb/integrity-reports-q1-2025/ (visited on 10/07/2025).

A

Legal Context

This appendix summarises the legal framework governing interception and processing of communications metadata in selected jurisdictions.

Summary by Jurisdiction

1. **European Union (ePrivacy & GDPR):** The e-Privacy framework (Directive 2002/58/EC[1] and related national laws) and data-protection rules under the GDPR regulate interception and processing of communications and metadata; interception without lawful basis or consent is prohibited. Processing of personal data requires a lawful basis (GDPR Art. 6[2]), purpose limitation and data minimisation.

2. **United States (Wiretap Act):** 18 U.S.C. § 2511 (the Wiretap Act)[3] generally prohibits interception of electronic communications without consent or statutory authority.

3. **Germany:** § 202a StGB[4] and related provisions criminalise unauthorised access to data; telecommunications secrecy is protected under §§ 88–109 of the Telecommunications Act (TKG)[5].

4. **Other jurisdictions:** National laws vary; many countries require judicial or statutory authorisation for interception and impose strict controls on retention and use of communications data.

[1]Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications).

[2]Regulation (EU) 2016/679 (General Data Protection Regulation), Article 6 (Lawfulness of processing).

[3]18 U.S.C. § 2511 — Interception and disclosure of wire, oral, or electronic communications prohibited.

[4]Strafgesetzbuch (StGB) § 202a — Unauthorised obtaining of data (Ausspähen von Daten).

[5]Telekommunikationsgesetz (TKG), Part 7 — Protection of privacy in telecommunications.

Practical Implication

Where the report states that `auth_key_id` is technically observable, that should be read as a statement of technical possibility. Any operational collection or retention of such identifiers requires legal authority and appropriate procedural safeguards; otherwise it risks criminal and civil liability.