A Systematic Literature Review of Secure Instant Messaging Applications from a Digital Forensics Perspective

ABDUR RAHMAN ONIK, Baggili(i) Truth (BiT) Lab, Center of Computation, Computer Science and Engineering, Louisiana State University System, Baton Rouge, United States
JOSEPH BROWN, Baggili(i) Truth (BiT) Lab, Center of Computation & Technology, Computer Science and Engineering, Louisiana State University System, Baton Rouge, United States
CLINTON WALKER, Baggili(i) Truth (BiT) Lab, Center of Computation & Technology, Computer Science and Engineering, Louisiana State University System, Baton Rouge, United States
IBRAHIM BAGGILI, Baggili(i) Truth (BiT) Lab, Center of Computation & Technology, Computer Science and Engineering, Louisiana State University System, Baton Rouge, United States

The use of instant messengers in organized crime has made retrieving digital evidence from these platforms crucial in investigations. However, the wide implementation of end-to-end encryption (e2ee) has made forensic analysis challenging since all data and multimedia stored in these platforms are encrypted. To retrieve evidence, forensic examiners rely on artifacts produced by secure messaging applications. Therefore, identifying the artifacts that can be gathered from secure messaging applications is crucial during emergency analysis, as law enforcement agencies monitor these platforms. This literature review aims to provide a comprehensive understanding of artifact retrieval and forensic analysis trends by summarizing studies conducted since inception on popular secure messaging applications, including WhatsApp, Signal, Telegram, Wickr, and Threema. The review covers artifact retrieval, forensic methodologies, and tools. The review findings are significant to the cybersecurity and research communities, as well as the users of these applications. Forensic investigators can leverage the information during their investigations, cybersecurity practitioners can identify weaknesses in implementation to develop better security policies, and users can make informed decisions regarding their privacy.

CCS Concepts: • **Security and privacy** → **Systems security**; • **Applied computing** → *Computer forensics*; • **Information systems**;

Additional Key Words and Phrases: Secure messaging application, WhatsApp, Signal, Telegram, Wickr, Threema, Forensic Investigation, Artifacts, End-to-End Encryption, CVEs, Vulnerability

1 Introduction

Instant messengers are popular applications for real-time or delayed communications that combine the immediacy of face-to-face meetings with the more deliberate pace of text-based conversations. These applications also offer unique features such as advertising user availability. They can be standalone web or mobile applications like Microsoft Teams or built-in functionality within other systems like Google Chat, which is integrated with Gmail.

Authors' Contact Information: Abdur Rahman Onik, Baggili(i) Truth (BiT) Lab, Center of Computation, Computer Science and Engineering, Louisiana State University System, Baton Rouge, Louisiana, United States; e-mail: aonik1@lsu.edu; Joseph Brown, Baggili(i) Truth (BiT) Lab, Center of Computation & Technology, Computer Science and Engineering, Louisiana State University System, Baton Rouge, Louisiana, United States; e-mail: jbro571@lsu.edu; Clinton Walker, Baggili(i) Truth (BiT) Lab, Center of Computation & Technology, Computer Science and Engineering, Louisiana State University System, Baton Rouge, Louisiana, United States; e-mail: cwal117@lsu.edu; Ibrahim Baggili, Baggili(i) Truth (BiT) Lab, Center of Computation & Technology, Computer Science and Engineering, Louisiana State University System, Baton Rouge, Louisiana, United States; e-mail: baggili@gmail.com.

ACM 1557-7341/2025/3-ART
https://doi.org/10.1145/3727641

Secure instant messengers are messaging applications designed from the ground up with security in mind. WhatsApp, Telegram, and Signal are the most widely used, with several billion combined users. WhatsApp is by far the largest, with nearly 2 billion active users. One of the smaller messengers, Telegram, still boasts 55 million active users [123]. Secure messengers intentionally obfuscate their communications, making them more difficult to intercept or retrieve messages from than traditional messengers. Their popularity and the difficulty of intercepting communication have attracted users with criminal intent alongside legitimate users. Cybercriminals televise, circulate, and execute various operations on secure messaging applications, eventually allowing them to steal credentials or other information from unwary users.

Secure Messaging (SM) applications employ various measures, including end-to-end encryption (e2ee), to secure data and traffic. Network traffic encryption makes it hard to obtain forensic information from intercepted traffic. These applications encrypt and store databases and multimedia files on mobile devices, making forensic examination difficult. This difficulty in retrieving artifacts may prompt law enforcement investigators to resort to alternative data acquisition methods, such as obtaining a warrant to compel companies to provide data. Differences in regional data regulations and privacy laws introduce additional obstacles.

Despite the growing use of secure messaging applications in various industries and personal communication, there has been no comprehensive literature review from the forensic perspective of these applications. As the reliance on secure messaging applications continues to increase, it becomes crucial to understand their security and potential vulnerabilities in the event of a criminal investigation. Our work is motivated by the need to fill this gap in the literature and to provide a deeper understanding of the forensic analysis capabilities of secure messaging applications.

In this study, we conducted a comprehensive analysis on the existing literature of the forensic analysis of secure messaging applications- WhatsApp, Signal, Wickr, Telegram, and Threema. We considered the user base of each application to ensure they were widely used and representative of the general population. The comparative lack of research on Wickr and Threema presented a unique opportunity to explore their research potential. To our knowledge, there are currently no surveys that cover secure messaging applications specifically from a forensic analysis perspective, further strengthening the importance and novelty of our study.

We aim to systematize the forensic work done on the selected secure messaging applications and identify research trends and changes in their security over time. We will analyze these findings and investigate the major challenges involved in the retrieval of secure messaging data. Our work makes the following contributions:

- Provides an integrated and synthesized overview of current research in the forensic analysis of secure messaging applications.
- Compares all research papers on secure messaging applications and analyzes the retrieval of artifacts, methodology, and forensic tools.
- Explains and analyzes the encryption techniques used by each application.
- This study provides Key Findings (KF) and identifies Research Gaps (RG) in the forensic analysis of secure messaging applications.[1]

The novel contributions of this work benefit not only forensic practitioners and security researchers but also the general public. Practitioners and researchers will gain insights into the methods used to retrieve artifacts from these applications and identify their weak points. The general public will be able to deepen their understanding of the true levels of security offered by these applications, each of which claims to be secure in some way. Additionally, the public will be able to access more information about the types of data these applications collect, store, and distribute. This knowledge will help guide end users to ensure their privacy.

This paper is organized as follows. Section 2 reviews related surveys, and Section 3 outlines the research methodology. Section 4 discusses forensic analysis of WhatsApp, Signal, Telegram, Wickr, and Threema, while

[1]Boxes containing RG and KF can be found throughout the paper where relevant.

Section 5 discusses secure messenger encryption protocols. Section 6 reviews published security vulnerabilities and attack vectors. The core research findings appear in Section 7, followed by legal and ethical considerations in Section 8. Section 9 provides further discussion and recommendations. Finally, Section 10 offers concluding remarks.

2 Related Surveys

This section provides a comprehensive review of existing surveys related to the forensic analysis of mobile applications. Our primary research objective is to conduct an in-depth investigation of secure messaging applications, including their encryption methodology, potential vulnerabilities, and security measures. Despite the fact that the papers reviewed in this section cover tangential topics to ours, this work represents the first comprehensive review of secure messaging applications from a forensic analysis perspective.

In 2010, Pilli et al. [99] conducted a survey on Network Forensics (NF), reviewing various analysis tools and proposing a generic process model. Ruan et al. [109] in 2011 aimed to gather insights about Cyber Forensics (CF), adding to the understanding of forensic analysis in network and cloud-based environments. In 2013, Kohn et al. [69] conducted a survey on Digital Forensics (DF) process models and proposed their own model, the Integrated Digital Forensic Process Model (IDFPM). Alamin and Mustafa [8] in 2015 surveyed tools and techniques used in the analysis of Android smartphones, while Harichandran et al. [57] conducted a survey to understand professional attitudes in CF, building on a 2004 survey by Rogers and Seigfried [107]. Faheem et al. [43] in 2016 presented a comprehensive review of forensic investigations in mobile devices and cloud environments. In 2018, Barmpatsalou et al. [22] surveyed the Mobile Forensics (MF) field to identify challenges and open issues. In 2019, Sanchez et al. [113] investigated the perspectives of practitioners working on Child Sexual Abuse Material (CSAM) cases. Lastly, in 2020, Al-Dhaqm et al. [7] reviewed Mobile Forensics Investigation Process Models (MFIPMs) and proposed an integrated model. All relevant survey papers are presented in Table 1.

Table 1. List of Survey Papers in the Field of Forensics

Paper	Topic	Contribution	Citations	Year
Rogers and Seigfried [107]	CF	Internet-based survey to identify top fives issues in CF	223	2004
Pilli et al. [99]	NF	Exhaustive survey on different NF frameworks and a new model was proposed	272	2010
Ruan et al. [109]	DF	Survey conducted among DF experts to find key aspects of cloud forensics, challenges, research direction	84	2011
Kohn et al. [69]	DF	Existing DF models from published research and tried to integrate into a unified forensic process model	205	2013
Alamin and Mustafa [8]	MF	Tools and techniques are used for android MF and comparative analysis of tools	7	2015
Faheem et al. [43]	MF	This comprehensive review discusses forensic investigations in mobile devices, cloud environments, and mobile cloud applications, addressing limitations and unique constraints encountered during such investigations.	40	2016
Harichandran et al. [57]	CF	Improving cyber forensics requires better education, cloud/mobile support, open-source tools, research, laws, and stakeholder communication.	84	2016
Barmpatsalou et al. [22]	MF	Reviewed 7 years of MF research, identifying gaps, presented shifts from past directions and addressed challenges and open issues	77	2018
Sanchez et al. [113]	DF	The study prioritized filtering technology on CSAM investigations, identified lack of research on Artificial Intelligence (AI) and Data Science (DS) techniques in investigations.	43	2019

Continued on next page

Table 1 – continued from previous page

Paper	Topic	Contribution	Citations	Year
Al-Dhaqm et al. [7]	MF	Analyzed MFIPMs, pinpointed current and future challenges, stating only a few models exist for a specific scenario.	30	2020
Johansen et al. [63]	SM	The paper compares six secure instant messaging mobile applications, testing 21 security and usability properties, and providing 12 recommendations for improvement.	7	2021
Andries et al. [17]	SM	This paper surveys the security protocols MTProto and Signal, used by popular messaging applications like Telegram and Signal, comparing their cryptographic primitives, end-to-end encryption features, and approaches to key management.	4	2022
Unger et al. [134]	SM	The paper evaluates and systematizes current secure messaging solutions, identifying three key challenges: trust establishment, conversation security, and transport privacy, and proposes an evaluation framework for their security, usability, and ease-of-adoption properties.	288	2015
Alatawi and Saxena [9]	SM	This paper surveys the security protocols MTProto and Signal, used by popular messaging applications like Telegram and Signal, comparing their cryptographic primitives, end-to-end encryption features, and approaches to key management.	5	2023

CF=Cyber Forensics, NF= Network Forensics, DF= Digital Forensics, MF= Mobile Forensics, SM= Secure Messaging

RG 1. Significant lack of detailed studies on the encryption mechanisms of SM applications in existing literature.

3 Research Methodology

To conduct a systematic literature review on secure messaging applications, we followed a four-stage process consisting of the following steps: search, collect, filter, and analyze [126].

3.1 Search

We initiated our literature search using broad-spectrum keyword queries across multiple academic databases, including Google Scholar, ScienceDirect, ACM Digital Library, and IEEE Xplore. Our initial queries encompassed various aspects of our research focus, such as “WhatsApp Forensic analysis,” “Signal Forensic analysis,” “Forensic analysis on the Social messaging application,”, “Wickr Forensic analysis,” “Threema Forensic analysis,” “Signal Forensic analysis,” “Telegram Forensic analysis,” “Forensic Survey Papers,” “ Secure messaging app encryption,” “Forensic analysis on the Social messaging application,” “Secure Messaging application vulnerabilities,” and “Secure messaging encryption”. As we progressed, we updated our keyword selection based on keywords from collected articles. We employed these additional keywords in subsequent searches to broaden our selections from the literature.

3.2 Collect

Using search query results, we collected articles from reputable online sources, including peer-reviewed journals and conference proceedings. We also gathered relevant news articles and blog posts from official sources of secure messaging applications. Our initial collection focused on titles and abstracts related to forensic analysis of secure messaging applications. We compiled these articles for detailed filtering and analysis. Our inclusion criteria gathered extensive surveys, research publications with proposed frameworks or implementations of forensic analysis techniques, and conceptual studies related to secure messaging applications.

3.3 Filter

We initially screened over 320 articles, examining abstracts, introductions, research questions, contributions, and conclusions. Our inclusion criteria focused on peer-reviewed papers, relevance to forensic analysis of secure messaging applications, and the presence of empirical data or significant theoretical contributions. Exclusion criteria involved duplicates, papers not directly related to forensic analysis, and those lacking substantial validation of forensic techniques. The remaining articles were then subjected to a thorough evaluation guided by our research questions, considering the forensic techniques employed, the quality of validation, and the significance of the findings.[2]

This meticulous screening approach resulted in the selection of 153 papers and articles for in-depth analysis. For ease of understanding and efficient referencing, we organized the literature into three categories: Technical Research Articles, Forensic Survey Articles, and News/Blog Posts. By categorizing studies, we ensure that they are reviewed comprehensively, adhering to the highest academic standards.

3.4 Analyze

During the analysis phase, we critically examined specific aspects of forensic analysis techniques presented in the surveyed literature. We developed a set of criteria through iterative discussions and refinement. These criteria encompassed:

- Identifying the current scope of forensic analysis techniques for secure messaging applications.
- Analyzing research trends over time based on technological advancements.
- Examining the specific secure messaging applications targeted by forensic analysis.
- Investigating the impact of encryption protocol evolution on forensic investigations.

We provided a thorough overview of the types of analysis conducted, the artifacts retrieved, the methodologies employed, the tools used, and the challenges encountered in forensic investigations into WhatsApp, Signal, Telegram, Wickr and Threema, as detailed in Table 4. Additionally, we offered a comprehensive examination of the locations of forensic artifacts of these secure messaging applications, as presented in Table 5 and comparative analysis for forensics tools used secure messaging application in Table 6.

> **KF 1.** Detailed mapping of forensic artifacts' locations within applications, including databases, multimedia files, logs, preferences, and account information, aids investigators in efficiently locating and extracting relevant data.

4 Literature Review

This section examines the literature on secure instant messenger application forensic analysis.

4.1 The WhatsApp Messenger Application

WhatsApp was developed by WhatsApp Inc. and currently boasts a user base of 2.24 billion individuals. The initial release of WhatsApp occurred in 2009 [96]. Throughout its history, WhatsApp has faced criticism due to persistent security vulnerabilities. This section will present a timeline of these security issues and provide an analysis of WhatsApp forensics. The first security flaw was identified in 2011, involving session hijacking of user accounts, and new issues have been steadily identified since.

In 2012, a security vulnerability in WhatsApp was exposed by Tweakers.net, enabling status alterations by unauthorized users [116]. Despite remediation efforts by WhatsApp, the vulnerability persisted, demonstrated by a released tool replicating the exploit [48]. The first comprehensive forensic analysis of WhatsApp, conducted by

[2]Notably, we included some blog posts and news articles that, despite not being peer-reviewed, provided strong technical insights with scientific validation.

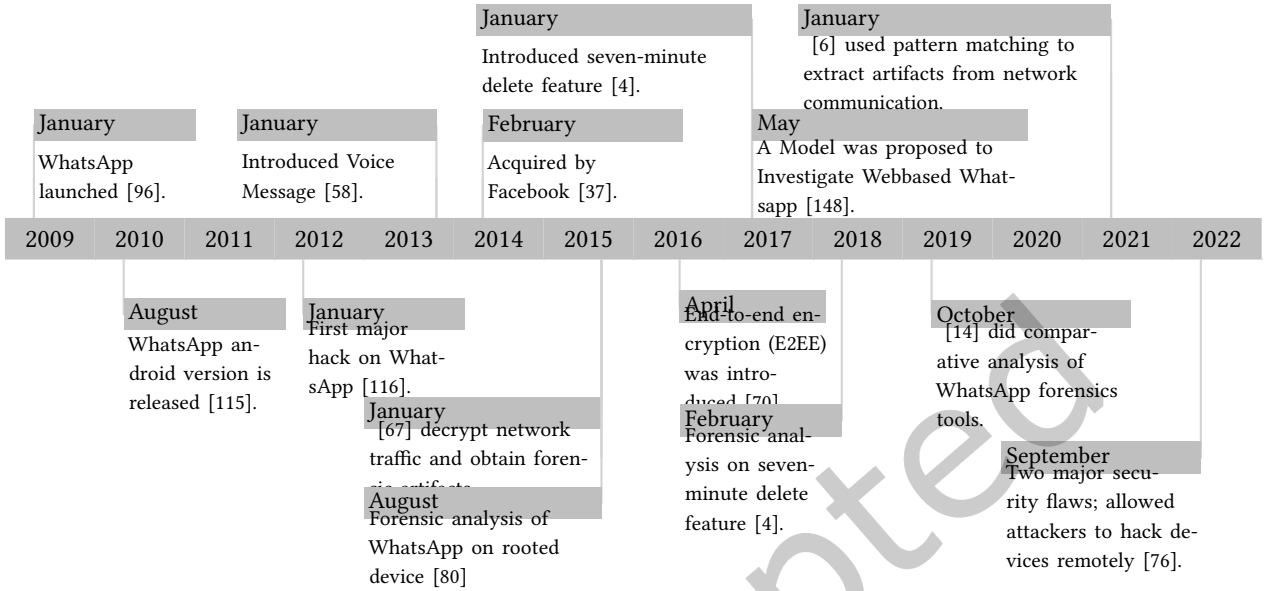

Fig. 1. Timeline of WhatsApp Milestones

Thakur [131], targeted Android devices, extracting user interactions from both volatile and non-volatile memory. Noteworthy artifacts like messages, contacts, locations, and deleted files were collected. In 2013, Mahajan et al. [83] utilized the Cellebrite Universal Forensic Extraction Device (UFED) to extract WhatsApp files, including contact lists, messages, media, and call details [83]. By 2014, Facebook acquired WhatsApp for $19 billion, aiming to expand its mobile messaging market presence and leverage WhatsApp's extensive user base, particularly in burgeoning internet and mobile usage regions [37]. In response to data privacy concerns, WhatsApp implemented robust database encryption methods [116].

In 2014, a vulnerability in WhatsApp's android encryption was discovered by Crook [35], enabling other apps to access chat conversations given permissions. Simultaneously, a flaw in location data transmission was identified [21]. Network protocol forensics gained prominence in 2015 when Hancke [56] investigated the WhatsApp call feature. Despite an initial inability to decipher call signaling messages, the researchers acquired phone numbers, timestamps, call duration, and server IP addresses from network traffic [56]. Enhanced security, including e2ee, was introduced in 2016 [116].

In 2017, a message deletion feature with a seven-minute window was introduced, although an anomaly with timestamps was identified [4]. Forensics on WhatsApp's web application occurred in 2018 [3, 93], revealing encrypted message storage. Subsequent research trends shifted to tool-based analysis, using forensic tools and frameworks such as National Institute of Standards and Technology (NIST), WhatsApp DB/Extractor, and BelkasoftEvidence Center (BEC) [132, 146].

In 2021, pattern matching was used to extract WhatsApp artifacts for crime detection [6]. Finally, two significant security flaws, Common Vulnerabilities and Exposure (CVE)-2022-36934 and CVE-2022-27492, in WhatsApp for android and iOS were discovered in 2022, allowing potential remote code execution [76].

Figure 1 presents a timeline illustrating significant milestones in WhatsApp research and security improvements, demonstrating the platform's ongoing commitment to addressing security concerns. These stats and this study show what WhatsApp is doing to secure user data and how vital it is to keep investigating.

4.2 The Wickr Messenger Application

Wickr, a free messaging application developed by Wickr Inc, is compliant with National Security Agency (NSA) security standards and was acquired by Amazon in June 2021 [123]. The application, which has surpassed 10 million downloads [61], supports e2ee across all communication forms and offers automatic deletion of messages and password-based messenger lock, among other security features. Wickr Me is the most popular version, with others including Wickr Pro, Wickr RAM, and Wickr Enterprise.

Wickr, assessed by the Electronic Frontier Foundation (EFF) with a score of 5 out of 7 [47], offers several security features. These include provider-inaccessible encrypted transit communications, identity verification, and forward secrecy, along with undergoing independent audits. Despite criticisms over its lack of open-source code and limited security design documentation, Wickr has published a white paper explaining its encryption protocol in 2021 [143].

Kim et al. [68] examined Wickr's encryption/decryption process, successfully decrypting data and verifying user passwords. Wickr utilizes AES256-GCM for core data encryption and stores the Core Data Key (CDK) based on user passwords. Exhaustive password recovery was deemed infeasible due to the Scrypt-implemented key derivation function (KDF), highlighting strong encryption's importance [68]. Wickr offers password and automatic login methods, the latter being default unless the messenger lock is activated. Son et al. [123] discovered an automatic login decryption algorithm that doesn't necessitate user passwords, instead extracting encryption keys from device files. Even with a password login switch, the algorithm remains effective due to persistent files used for automatic login and consistent database encryption key generation [123].

Wickr's robust security features, including self-destruct timers and encryption, position it as a leading secure messaging application. Its anti-forensic features and encryption protocol add to its security, making artifact collection challenging for investigators. In December 2023, Wickr is transitioning to the Amazon Web Services (AWS) platform, AWS Wickr. This shift is expected to further enhance security and scalability, providing users with an even more reliable messaging experience while maintaining the high standards of privacy and security associated with the Wickr brand [125].

While available forensic analysis on Wickr is limited compared to other messaging applications, existing research has provided a glimpse into the inner workings of Wickr's security measures. In general, the findings demonstrate Wickr is a privacy-focused messaging application presenting challenges for forensic analysis. As the landscape of secure messaging applications continues to evolve, with Wickr's move to AWS being a notable example, future research and forensic investigations will need to adapt to these changing paradigms to maintain an accurate understanding of the security and privacy offered by these services.[3]

4.3 The Telegram Messenger Application

Telegram, a feature-rich instant messaging application, is known for its cloud-based technology and security features such as end-to-end encrypted chat and file sharing. Telegram has over 440 million downloads and 55.2 million daily users [110]. Telegram's secret chat feature, which uses client-to-client encryption, is considered more secure [121].

Telegram's MTProto encryption protocol has had vulnerabilities, including a man-in-the-middle attack [129], and the application has been rated as "problematic" by Stiftung Warentest, a German consumer organization [138]. The EFF rated Telegram's secret chat highly for security, while its default chat feature scored lower due to issues such as identity confirmation and encryption key security [47]. In 2015, Telegram rolled out the channel feature, restricting message posting to administrators and facilitating large scale broadcasting [79]. Around the same time, Aarhus University researchers found vulnerabilities in Telegram's MTProto 1.0 encryption protocol [62].

[3]As of December 2023, Wickr is no longer available for individuals. Amazon is focusing on maintaining and supporting business users through AWS Wickr. It remains important to analyze Wickr for forensic purposes due to its robust security features and historical usage patterns.

Telegram assured users the issues did not compromise security and subsequently introduced MTProto 2.0 in 2017 for enhanced security compliance [34, 133].

Satrya et al. [114] have extracted data from the Telegram android system for potential use in criminal cases. Gregorio et al. [54] developed a method for analyzing Telegram on Windows mobile devices. Borodin et al. [28] simulated Telegram artifacts on Windows. Wicaksanaa and Suhartanaa [141] performed a crime simulation on the desktop application. In these studies, researchers have successfully recovered user messages, chat history, and other encrypted data. The MTProto protocol, despite early vulnerabilities, has been found to offer secure communication, with Telegram quick to address identified flaws [10, 11]. Notably, a 2021 discovery of a self-destruct function vulnerability led to the recovery of deleted images [119]. Recent studies in 2022 have furthered Telegram's forensic analysis, with Raza and Hassan [105] examining the application in an Android virtual environment, while Fernández-Álvarez and Rodríguez [46] performed an exhaustive forensic analysis on Telegram Desktop. These studies continue to provide valuable insights into potential cybercrime investigations.

4.4 The Signal Messenger Application

Signal Messenger Application is an open-source secure messaging application that has gained popularity for its focus on user privacy and security. The application, founded in 2014, uses end-to-end encryption to ensure every communication is secure and confidential [84]. It has 40 million active users monthly and has been downloaded over 125 million times [36]. Signal, in addition to text messaging, enables phone and video calls and the ability to establish and join groups. The application is also available for Windows, macOS, and Linux desktop computers. Signal has been audited by independent researchers and has received positive evaluations of its security and privacy features. Signal is considered one of the most secure instant messaging applications, making it a valuable tool for secure communication in various fields such as enterprise, personal, and government. In January 2021, rival WhatsApp introduced an unpopular new privacy policy, which was communicated to users through a splash screen. In the wake of this update to WhatsApp, Signal experienced a surge in new users [52].

Present studies aim to evaluate Signal's features and security protocols. Signal allows for creating user accounts utilizing a cellular telephone number, including those from providers such as Google Voice. The application utilizes E2EE for all forms of communication, including text messages, audio, and video calls, group chats, and shared media. Of particular note is the utilization of the Signal Protocol (previously known as the TextSecure Protocol) [147] for encryption, which has been widely recognized for its robust security.

Judge [64] conducted mobile forensics on the Signal application to evaluate the feasibility of obtaining forensic artifacts and to determine which tools and methods most effectively uncover evidence. The researchers employed a UFED and the UFED Physical Analyzer as forensic instruments, and they compared Cellebrite's results with the results generated by three open-source tools: iExplorer, Autopsy, and iPhone Analyzer. The findings varied by device make and model, and only the ZTE Android device (Z993 Prelude) allowed physical extraction among the four devices examined.

In 2019, Kaczyński [66] conducted a security analysis of Signal's Android database. Despite extensive examination, they found no usable forensic artifacts, and all attempts to decrypt the database were unsuccessful. However, the study did reveal that Signal users remain vulnerable to multiple security threats—primarily due to the lack of any additional safeguards beyond those provided by the operating system. In particular, it is possible to reconstruct a conversation's history and impersonate one of the communicants, and because standard identifiers remain unchanged, the other party would be unaware of the attack. This vulnerability applies to all Android systems, regardless of whether the secret key used for database encryption is stored in the system KeyStore. In response, the authors advised restoring access passwords in Signal's Android application. Subsequently, in 2020, Azhar et al. [20] employed Oxygen Forensic Detective (OFD) for Android to perform a forensic examination of Signal on a rooted Android phone but were again unable to retrieve any conversation or account information.

In 2022, Son et al. [123] introduced a new approach to analyzing Signal's decryption algorithm, gathering static and dynamic data from both unrooted and rooted smartphones. They successfully extracted the decrypted database, multimedia, log files, and preferences, enabling investigators to reconstruct user activities, interests, events, and preferences. These findings provide significant value for forensic investigators, and the researchers additionally developed an application to retrieve keys from the Android Keystore for deeper analysis.

RG 2. Cellebrite, a tool often used by law enforcement to gather evidence, claims that it can extract data from Signal [32]. However, Signal has responded by pointing out weaknesses in the Cellebrite system [85]. Despite this exchange, there is still little research on whether Cellebrite can effectively be used for Signal forensics. This unexplored area could provide an exciting opportunity for future research.

4.5 The Threema Messaging Application

Threema is an open-source secure messaging application developed and distributed by Threema GmbH. The German consumer group Stiftung Warentest ranked Threema as the second best out of 18 messenger applications tested in 2015 [123]. Threema has a significant user base, with over 10 million users in Switzerland alone. Furthermore, over one million downloads of the application have been recorded on the Google Play Store [65].

The application features e2ee for all forms of communication, including group texts, photographs, videos, files, and voice calls. This encryption is achieved through Advanced Encryption Standard (AES-256), ensuring the protection of stored communications and media. Additionally, Threema offers the option to lock messages for added security. AES-256 is a symmetric encryption algorithm that protects sensitive data [100]. The encryption key used in AES-256 is 256 bits long, making it difficult to crack through brute-force attacks [104]. The encryption process is designed to protect stored communications and media against unauthorized access, making messaging a more secure method.

Forensic analysis of Threema is limited to the work of Son et al. [123], where data was extracted from both unrooted and rooted devices, and static and dynamic analysis was performed to determine the decryption algorithms used by the application. The study was successful in decrypting databases and multimedia files. The study found the package name which is *ch.threema.app*. Two database files were found in the messenger's internal and external storage. The encrypted database file was named $threema4.db$, while another database file $threemanonce-blob4.db$ was not encrypted. Binary data files were found in the $INTERNAL/files/$ folder and multimedia files were encrypted and stored as hidden files in $EXTERNAL/Android/data/ch.threema.app/files/data/$. The cache data was found in the *cache/image_manager_disk_cache* folder. The *key.dat* and *msgqueue.ser* files' data are changed when the messenger lock feature is used.

In 2022, Paterson et al. [97] identified seven potential vulnerabilities in Threema's custom-designed encryption protocol. Two significant issues could permit attackers to impersonate a user without needing special access. One vulnerability lies in the ephemeral keys used for user authentication, as exposure could lead to misuse by an attacker. The other relates to a flaw in the client-to-server protocol interaction with encryption, which could inadvertently enable the attacker to impersonate the user. Three other identified issues require server access. These encompass a lack of metadata messages protection, enabling message reordering; susceptibility to reflection and replay attacks, leading to old or duplicated message resending; and issues within the challenge and response protocol flow for user-server authentication. The final two vulnerabilities emerge if an attacker gains access to an unlocked phone. Given Threema allows for private key exportation to new devices, an account could be cloned without a Threema account pin or password. Moreover, a bug that facilitates message backup before encryption could enable attackers to infiltrate the backup and recover the private key. Threema, however, has dismissed these findings as theoretical, claiming no real-world implications.

In conclusion, Threema's appeal, particularly in Switzerland, stems from its use of end-to-end AES-256 encryption and secure messaging. Future research should focus on several Threema-related aspects. Comparative studies with other popular instant messaging applications would help evaluate Threema's privacy and security strengths. Further investigation into AES-256 and other encryption algorithms would illuminate the most secure communication methods. The forensic analysis could reveal potential weaknesses in Threema's data storage and encryption key management. Lastly, understanding Threema's user experience and adoption could highlight factors driving its popularity and potential improvement areas.

We have undertaken an extensive analysis of security research focused on WhatsApp, Wickr, Signal, Telegram, and Threema. Particular emphasis has been placed on exploring their features, security aspects, user bases, and the volume of research literature focused on each. WhatsApp, given its significant user base and the extensive amount of research around it, has been treated separately, with its milestones and critical aspects illustrated in Fig 1. For the rest of the applications - Wickr, Signal, Telegram, and Threema - we have compiled a comprehensive comparison that is presented in Table 2. This table encapsulates the key milestones for each platform, providing a succinct yet detailed overview.

KF 2. Secure messaging applications often do not prioritize usability in their security features. Interfaces designed for technical robustness may not translate well to everyday use. Conducting behavioral studies to understand how users interact with security features in real-life scenarios can provide insights into improving usability.

KF 3. Memory analysis is underutilized for capturing encryption keys and session data in secure messaging, which are crucial for decrypting messages.

KF 4. Data stored on mobile devices by secure messaging applications is volatile. Forensic investigators must recognize the risk of data loss and implement strategies to safeguard the evidence.

Table 2. Milestones of Wickr, Telegram, Signal, Threema

Milestone	Wickr	Telegram	Signal	Threema
First Release	For iOS, the first release was in 2012 [123].	Released in 2013 [130].	Released in 2014 [84].	Released in 2012 [51]
EFF Score	In 2015, EFF scored Wickr 5 out of 7 [131].	In 2015, EFF scored Telegram 4 out of 7 for chat and 7 out of 7 for a secret chat [29].	In 2014, EFF scored Signal 7 out of 7 [47].	In 2016, EFF scored 6 out of 7 [47].
Userbase	10+ Million [95].	55.2 Million [95].	40 Million [95].	10 Million [95]
Forensic Research	First forensic analysis in 2013 [87].	First major security issue in 2013 [120].	Judge [64] did first forensics on Signal.	Rösler et al. [108] analyzed group communication protocol.
	In 2014, the desktop version with multiple device syncing was released [87].	First forensic analysis paper in a peer-reviewed journal in 2016 [45].	First iOS application to enable e2ee voice calls [81].	Son et al. [123] did first forensic.

Table 2 – continued from previous page

Milestone	Wickr	Telegram	Signal	Threema
	Forensic analysis of Wickr, the study could not recover any artifacts [132].	The first forensic analysis of the telegram on the Windows phone was in 2017 [101].	In March 2017, the sergeant at arms of the United States Senate approved Signal for usage by senators and their employees [140].	
	In 2017, published a paper for its source code and released messaging protocol on GitHub [95].	First forensic analysis of macOS in 2018 [149]. In 2020 forensic analysis on Telegram desktop [141].	Introduced Signal Protocol in 2018 [82].	
	As of 2021, it was regarded as one of the most anti-forensic messaging applications [6].	Forensic of telegram messenger application in an android virtual environment in 2022 Kunang and Khristian [74].	In 2019, Google's Project Zero team identified a flaw in the Android Signal client that could secretly allow spying [71].	

RG 3. There is a lack of research on real-time forensic analysis to capture data as it is transmitted, which is crucial for intercepting communications during active investigations.

5 End to end encryption of Secure Messaging Applications

E2EE, the foundation of secure communication, ensures that only the sender and receiver can access the data [41]. Devices encrypt messages before transmission and only the intended recipients can decrypt them. Private keys, crucial for decryption, reside solely on user devices. This utilization of public and private keys for encryption and decryption, respectively, is known as asymmetric cryptography. E2EE generates unique public and private cryptographic keys for each participant, enhancing security [41].

To fully understand the security mechanisms employed by secure messaging applications, it is essential to grasp the fundamental cryptographic primitives they utilize. The following are brief definitions of these key cryptographic primitives:

AES-256: Symmetric encryption algorithm used in secure messaging to protect data confidentiality and integrity. It operates on 128-bit data blocks using a 256-bit key, undergoing 14 rounds of encryption involving substitution, permutation, and mixing [44, 103]. For example, when Alice sends a message to Bob, they agree on a shared 256-bit key. Alice encrypts her message in 128-bit blocks, resulting in ciphertext, which she sends to Bob. Bob then uses the same key to decrypt the ciphertext back into the original message. This process ensures that only Alice and Bob can read the message, providing robust security against brute-force attacks.

Hash-based Message Authentication Code (HMAC-SHA256): Cryptographic technique used in secure messaging applications to ensure both data integrity and authenticity. It combines the SHA-256 hash function with a secret key shared between the sender and receiver. The process involves creating an inner and outer hash using the key, message, and specific padding values, resulting in a 256-bit hash value [38]. For instance, when Alice sends a message to Bob, she computes an HMAC-SHA256 hash using their shared secret key and appends it to the message. Upon receipt, Bob generates his own HMAC-SHA256 hash with the same key and compares it to the received hash. If they match, Bob can be confident that the message is authentic and unaltered.

Elliptic Curve Diffie-Hellman (ECDH): key exchange protocol that allows two parties to generate a shared secret over an insecure channel using elliptic curve cryptography [73, 127]. Each party generates a private-public key pair and exchanges their public keys. The shared secret is computed using the private key of one party and the public key of the other, providing a secure basis for encrypting subsequent communications.

Double Ratchet Algorithm: Cryptographic protocol which combines public-key cryptography with symmetric-key cryptography to provide forward secrecy and future secrecy. The algorithm continuously updates encryption keys with each message sent, ensuring that even if one key is compromised, past and future messages remain secure [16, 98].

Secure Hash Algorithm 256-bit (SHA-256): Cryptographic hash function that produces a 256-bit fixed-size hash value from input data. It is designed to be collision-resistant, meaning it is computationally infeasible to find two different inputs that produce the same hash value [49]. SHA-256 is used for data integrity and verification in security protocols [55].

Diffie-Hellman (DH) Key Exchange: A method for two parties to securely share a common secret key over an insecure communication channel [78]. Each party generates a public-private key pair and exchanges the public keys. The shared secret is computed using the private key of one party and the public key of the other, forming a basis for secure communication [26, 78].

XSalsa20: XSalsa20 is a stream cipher that uses a 256-bit key and a 192-bit nonce to encrypt data. It is an extended version of the Salsa20 cipher, designed to provide high-speed encryption while maintaining security. XSalsa20 is known for its resistance to cryptographic attacks [77].

Poly1305: Poly1305 is a cryptographic message authentication code designed to provide high-speed, high-security authentication. It generates a 128-bit tag that verifies the integrity and authenticity of a message. Poly1305 is often used in conjunction with stream ciphers like XSalsa20 to ensure both encryption and authentication [24].

Password-Based Key Derivation Function 2 (PBKDF2): PBKDF2 is a key stretching algorithm that uses a password, a salt, and multiple iterations of a hash function to generate a cryptographic key [42]. This process makes it computationally expensive to perform brute-force attacks, enhancing the security of stored passwords [135].

scrypt: scrypt is a key derivation function designed to be computationally intensive and memory-hard, making it resistant to hardware-based brute-force attacks [33]. It is used to securely derive cryptographic keys from passwords, offering strong protection against password cracking [15].

Networking and Cryptography library (NaCl): Cryptographic library that provides high-speed, secure encryption, decryption, and authentication functions. It includes primitives like XSalsa20 for encryption and Poly1305 for message authentication, designed to offer robust security with efficient performance [25].

Following subsections delve into the specifics of e2ee as deployed in WhatsApp, Telegram, Signal, Wickr, and Threema, and also cover a thorough evaluation of security and privacy. The detailed descriptions and diagrams of the encryption processes for each system were derived primarily from the official whitepapers and technical documentation provided by the respective developers.

5.1 WhatsApp End-to-end Encryption

WhatsApp employs an e2ee protocol, derived from Open Whisper Systems' Signal Protocol [40]. The encryption process bars third-party access to communications, ensuring privacy for calls and messages. To register a WhatsApp account, a user's phone number is needed, designating the device as the primary one. Linking to additional devices, termed companion devices, is possible, although iPhone and Android platforms currently lack support for functioning as companion devices to other primary accounts.

During registration, three types of public keys are created and stored on WhatsApp's servers: Identity Key Pair, Signed Pre Key, and One-Time Pre Keys. The Identity Key is vital for identification and trust establishment during encryption, while the Signed Pre Key, authenticated by the Identity Key, helps initiate encryption sessions.

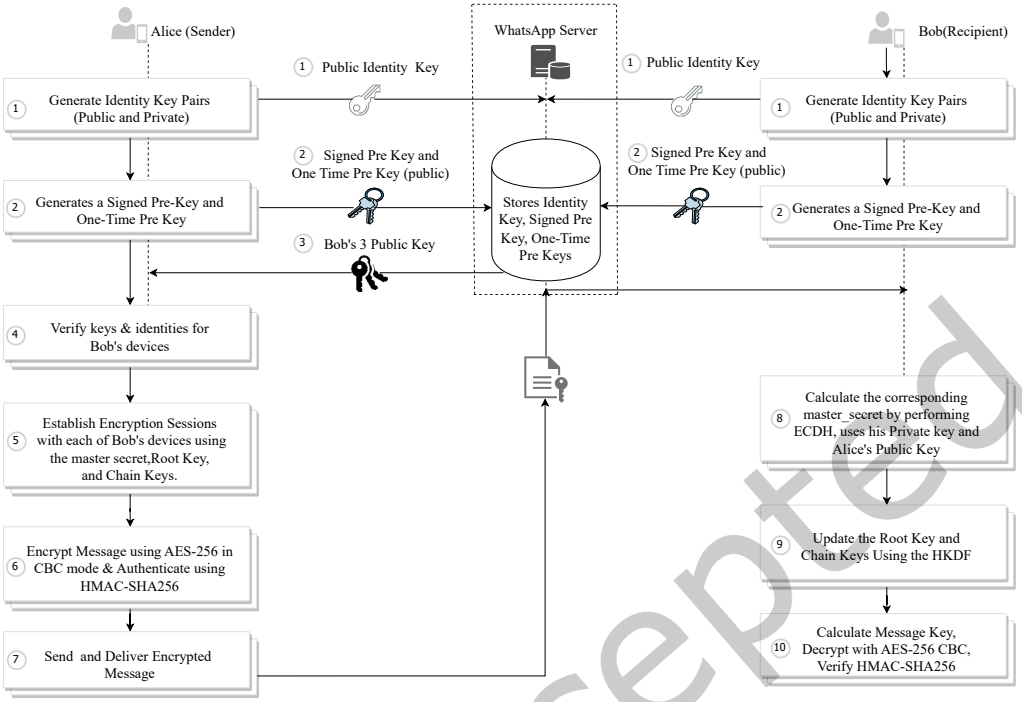

Fig. 2. WhatsApp End-To-End Encryption

One-Time Pre Keys are one use public keys maintaining forward secrecy, ensuring past communication stays secure even if future private keys are compromised.

Three session keys are used to secure communication. The root key (32-byte) is generated in the initial session setup and updated each time one message cycle is completed between a sender and a receiver. It helps to maintain forward secrecy and creates Chain Keys. Chain keys are 32-byte values used to create message keys. Each message generates a new Chain Key from the current Chain Key, making it difficult to retrieve previous or future Chain Keys.

Message keys are used to encrypt the message content. It comprises a 32-byte AES-256 key, a 32-byte HMAC-SHA256 key, and a 16-byte Initialization Vector (IV). The Message Key is ephemeral- it cannot be reconstructed from the session state after a message has been sent or received. This provides additional security, as each message is encrypted using a unique key.

Figure 2 depicts WhatsApp e2ee and the following is an outline of how WhatsApp sets up its encryption and how it happens during a conversation:

(1) At registration, Alice and Bob each generate an Identity Key Pair (public and private). They send their public keys to the WhatsApp server.
(2) Alice and Bob each also generate Signed Pre keys and One-Time Pre keys and send them to the WhatsApp server. The server stores these with the Public identity key.
(3) When Alice wants to send a message to Bob, she requests Bob's public keys (Identity Key, Signed Pre Key, and One-Time Pre Key) from the server. The server also returns any additional public keys for Bob's linked companion devices.

(4) When Alice's client receives the public keys for each of Bob's devices, including any companion devices, it needs to verify the keys and identities to ensure the keys are authentic and belong to the intended recipient (Bob) before establishing encryption sessions. This step helps prevent potential man-in-the-middle attacks or other attempts to compromise communication. For each set of public keys (Identity Key, Signed Pre Key, and One-Time Pre Key) associated with Bob's devices, Alice's client first checks the signature on the Signed Pre Key using Bob's Identity Key. This confirms that the Signed Pre Key genuinely belongs to Bob.
(5) For each of Bob's devices, Alice establishes a separate encryption session. Alice calculates a master secret using her private keys and Bob's public keys. From this master secret, the client derives a Root Key and Chain Keys using the HMAC-based Extract-and-Expand Key Derivation Function (HKDF) function.
(6) Alice generates a Message Key from the Chain Key for each device. The message is then encrypted using AES-256 in Cipher Block Chaining (CBC) mode and authenticated with HMAC-SHA256.
(7) Alice sends the encrypted message to each of Bob's devices. The server delivers the message to Bob's devices, which can decrypt the message using the established encryption session.
(8) Once Bob receives the message, Bob uses its own private keys and the public keys included in the message header to compute the corresponding master_secret. This is done by performing ECDH operations with the appropriate key pairs.
(9) Using the HKDF, Bob derives a new Root Key and Chain Key from the master_secret and ephemeral_secret (obtained from the ECDH operation with the ephemeral keys). This ensures forward secrecy, as updating the keys makes it difficult for an attacker to decrypt past or future messages even if they manage to compromise a single key.
(10) With the updated keys, Bob calculates the Message Key from the Chain Key using the HMAC-SHA256 function. This Message Key is then used to decrypt the message content, which was encrypted using AES-256 in CBC mode.

During the exchange, Alice and Bob constantly update their Chain Keys, providing forward secrecy in case an attacker compromises any of their keys. In end-to-end encrypted chats, additional information about the sender and recipient's devices is included inside the encrypted payload, ensuring consistency across devices.

The Signal protocol ensures that a WhatsApp conversation is safe by combining long-term and temporary key pairs, prekeys, Root Keys, Chain Keys, and Message Keys. The forward secrecy provided by the ratcheting mechanism makes it difficult for someone to access or decrypt past messages. The ratcheting mechanism constantly updates encryption keys, ensuring that even if an attacker obtains a user's encryption key, they cannot use it to decrypt past messages.

5.2 WhatsApp Security and Privacy Evaluation

Our study raises concerns about metadata security in WhatsApp. Although metadata is encrypted during transmission, WhatsApp servers retain data such as phone numbers, timestamps, call length, frequency, and user location, enabling investigators to develop detailed profiles [102]. Furthermore, while WhatsApp messages are securely transmitted, many devices, unlike Apple's latest iPhones [102], do not encrypt stored data. This is also true for WhatsApp backups on cloud servers, like Google Drive or Apple iCloud, where the encryption status remains unclear. Physical extraction of data from devices may also be possible using specialized tools or live acquisition techniques if the device is unlocked. Additionally, the use of third-party services to manage API endpoints in WhatsApp could undermine the validity of end-to-end encryption, as these vendors might access message content, posing a potential security risk [60].

5.3 Wickr End-to-End Encryption

In a digital age where information is transmitted constantly, ensuring messages are protected through robust encryption methods is crucial to preventing unauthorized access and safeguarding sensitive data. While end-to-end encryption is crucial, it is only part of the tale. The true difficulty is in safely disseminating the encryption key. Data is encrypted using a secret key, which is then used to decipher the ciphertext on the receiving end. But what if somebody manages to snoop on this encrypted key as it travels from sender to receiver? Then even the most complex encryption algorithm applied to the messages will be easily unlocked. Based on this context Wickr applied multiple layers of encryption. Wickr uses 256-bit authenticated end-to-end encryption [59]. In Wickr's e2ee, the following scenario occurs when Alice sends a message to Bob (See Figure 3):

(1) Alice and Bob each have an ECDH-521 key pair consisting of a public key and a private key. The public key is shared with other users, while the private key is securely stored on the user's device.
(2) Alice generates a unique AES-256 key for the message.
(3) Wickr combines this unique message key with Alice's message to generate a scrambled message. Wickr not only encrypts the message, it also encrypts the encryption key.
(4) Alice computes a shared secret using her ECDH private key and Bob's ECDH public key.
(5) Alice derives a symmetric encryption key from the shared secret using a KDF. Alice encrypts the message-specific AES key using the symmetric encryption key derived from the shared secret. Wickr implements Perfect Forward Secrecy (PFS) here, which means a unique encryption key for each message, ensuring that even if a single key is compromised, it cannot be used to decrypt other messages in the conversation. PFS helps protect messages from future attacks or attempts to decrypt the data retrospectively. Encryption keys are securely managed on user devices using cryptographic key management techniques. Wickr stores the keys in a secure storage area on the device, where they are protected by the device's built-in security features and encryption mechanisms. Access to these keys is controlled and limited only to authorized applications like Wickr.
(6) Encrypted message and Encrypted Key sent to Wickr Server.
(7) Server added security Layer using TLS.
(8) Bob computes the same shared secret using their ECDH private key and the sender's ECDH public key.
(9) Bob decrypts the Advanced Encryption Standard (AES) key using the symmetric encryption key derived from the shared secret.
(10) Finally, Bob uses the decrypted AES key to decrypt the encrypted message content.

5.4 Wickr Security and Privacy Evaluation

The Wickr platform stores some metadata to maintain system continuity and improve user experience [118]. The metadata Wickr stores does not contain messages or any other information that can be linked with the message delivery. Wickr stores hashed user and device information, but those can not be linked back to the user. Metadata related to the sender and receiver are stored only for the time it takes to deliver the messages and then deleted from the server.

5.5 Telegram End-to-End Encryption

Telegram developed and uses MTProto 2.0 to fulfill its platform's requirements. Figure 4 depicts Telegram e2ee. It uses symmetric encryption, meaning the same encryption key is used for both encryption and decryption. An authorization key is generated by both the client and the server in order to establish secure communication. A DH key exchange protocol generates this key, so the key remains confidential even if the communication channel is compromised. Using AES in infinite garble extension mode, all messages are encrypted when the **auth_key** is generated, ensuring their security during transit. Before transmission through the network, the message is

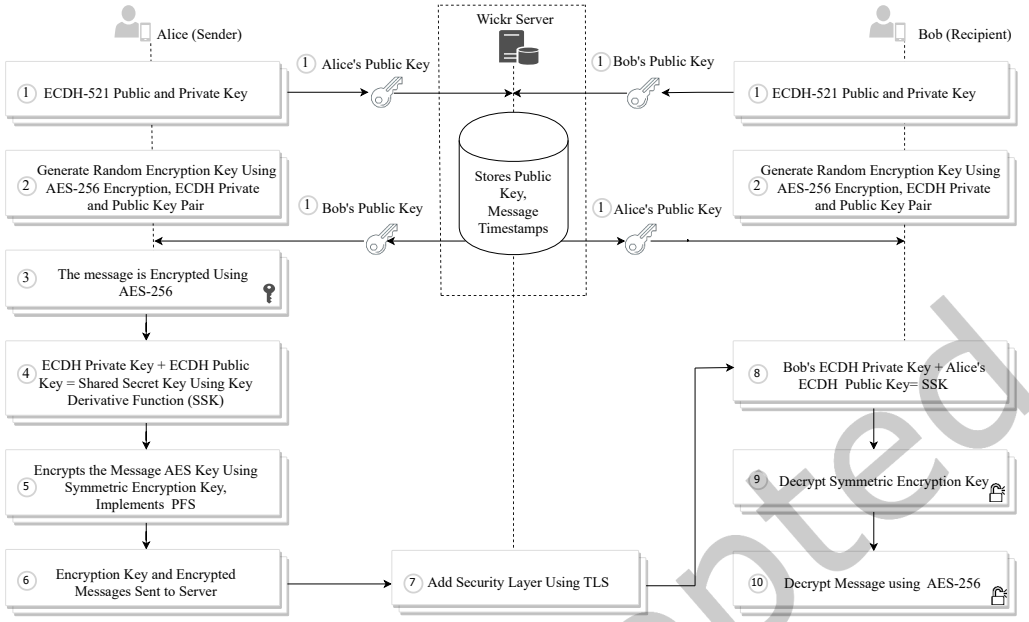

Fig. 3. Wickr End-To-End Encryption

encrypted with an external header. This header is a combination of two components: a 64-bit key identifier called **auth_key_id** and a 128-bit message key name **msg_key**. This **auth_key_id** is a unique identifier generated by hashing the **auth_key** using the SHA-1 algorithm and extracting the least significant 64 bits of the resulting hash, both server, and the user. The **msg_key** is the 128-bit value used as an additional layer of security. It is generated from the portion of encrypted messages using the SHA-256 hash function, which helps to ensure that messages are not altered or tampered with during transit.

Below is a step-by-step explanation of how end-to-end encryption works in Telegram when Alice sends a message to Bob (See Figure 4):

(1) When Alice and Bob decide to start a secret chat, their devices independently generate a set of public and private encryption keys using the DH key exchange protocol. Alice's device creates a public key (**A_pub**) and a private key (**A_priv**), while Bob's device creates a public key (**B_pub**) and a private key (**B_priv**). Alice's and Bob's devices exchange their public keys, **A_pub** and **B_pub**. This exchange is done securely through Telegram's servers, but the private keys (**A_priv** and **B_priv**) are never shared and remain only on their respective devices.
(2) Both devices now compute a shared secret key using the exchanged public keys and their private keys. Alice's device computes the shared secret as S = (**B_pub**) raised to **A_priv**, and Bob's device computes the shared secret as **S** = (**A_pub**) raised to **B_priv**. Since these calculations are equivalent, both devices obtain the same shared secret key, **S**.
(3) The shared secret key, **S**, is then passed through a KDF to create a symmetric encryption key, which is used to encrypt and decrypt messages in the secret chat. The KDF ensures the derived key has suitable cryptographic properties.

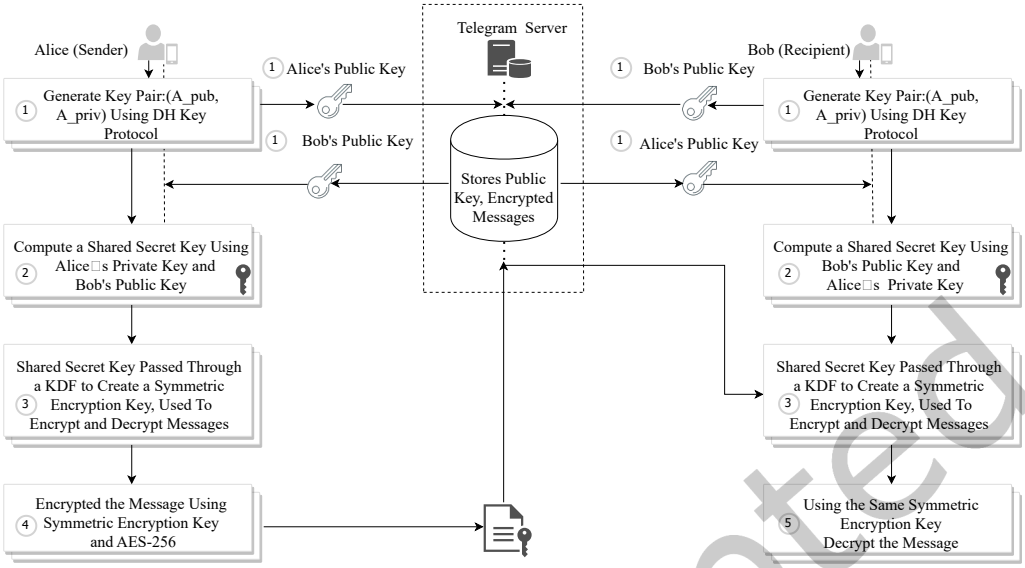

Fig. 4. Telegram End-To-End Encryption

(4) When Alice sends a message to Bob, her device encrypts the message using the derived symmetric encryption key and an encryption algorithm (e.g., AES-256). The encrypted message is sent through Telegram's servers to Bob's device.
(5) Upon receiving the encrypted message, Bob's device uses the same symmetric encryption key to decrypt the message and reveal the original text. Since only Alice and Bob's devices have the shared secret key, no one else, including Telegram, can decrypt the message.

To further enhance security, Telegram uses perfect forward secrecy. This means that new encryption keys are generated for each message or periodically during the conversation. If a key is compromised, only a small portion of the chat is at risk.

5.6 Telegram Security and Privacy Evaluation

An attacker may be able to retrieve messages through a man-in-the-middle (MITM) attack by intercepting key exchanges during the encryption process. By doing so, the attacker could gain access to the shared secret key (**auth_key**) and decrypt the intercepted messages. In this type of attack, hackers place them between two parties and intercept the public value exchanged during DH key exchange protocol. The hacker generates their public-private key pair and forwards the public key to both users. Each user thinks their communication is secure and shares a secret key with the hackers. The hackers can decrypt messages then and potentially modify these messages.

5.7 Signal End to End Encryption

Signal employs the Signal Protocol for implementing e2ee, ensuring robust security. This protocol incorporates perfect forward secrecy, safeguarding past and future messages even if a decryption key is compromised. It also supports asynchronous operation, allowing offline recipients to decrypt messages upon availability. Signal's e2ee

extends to all communication types, including text, voice, and video calls, and file transfers, contributing to its distinction as a highly secure messaging application.

Figure 5 represents how message encryption works when Alice sends a message to Bob.

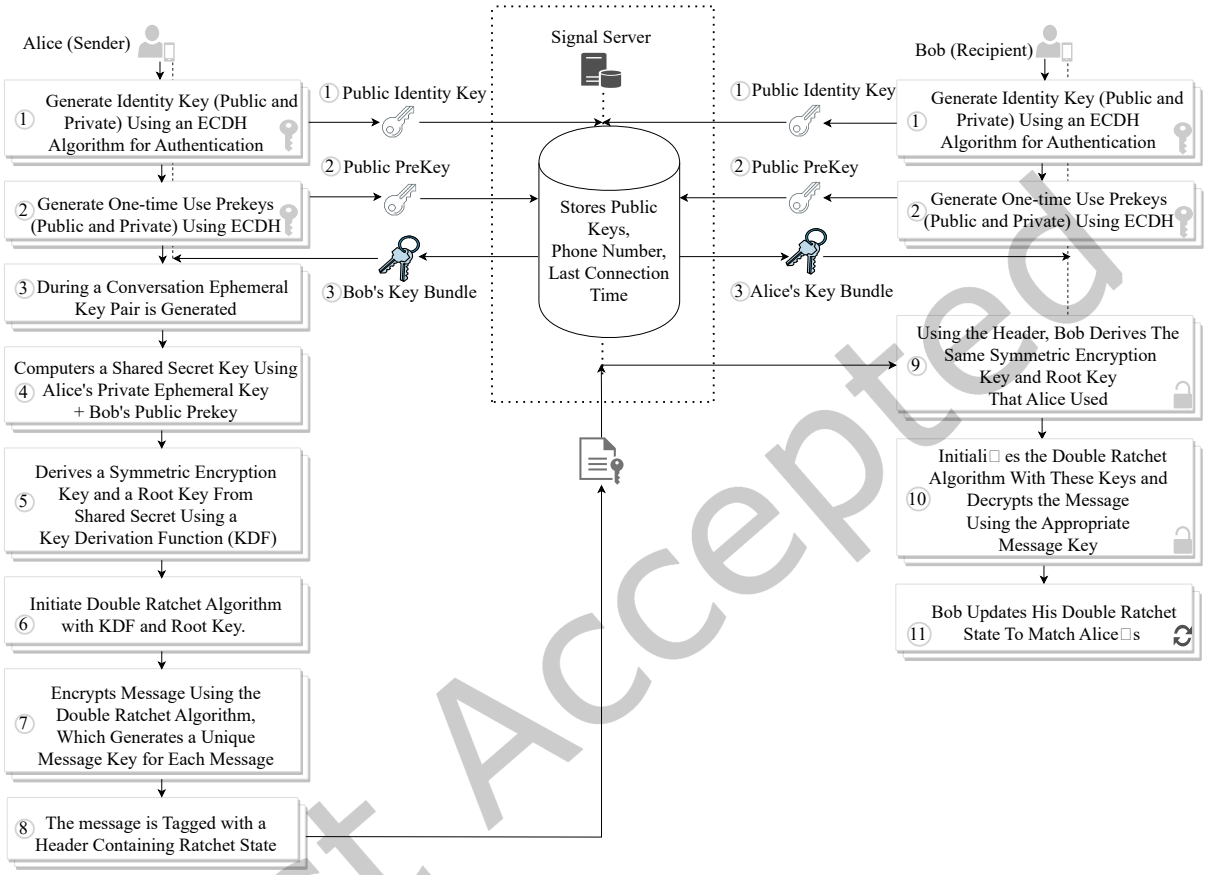

Fig. 5. Signal End-To-End Encryption

(1) Alice and Bob generate long-term identity keys (public and private) using an ECDH algorithm. These keys are used to authenticate each other. The public identity keys are sent to the Signal server.
(2) Alice and Bob generate a set of one-time-use prekeys (public and private) based on ECDH. The public prekeys are sent to the Signal server.
(3) When Alice wants to send a message to Bob, she requests Bob's prekey bundle from the Signal server, which includes Bob's public identity key, public prekey, and a signed prekey signature. Bob has a prekey pair (public and private prekeys). Bob uses his private identity key to sign the public prekey, creating the signed prekey signature. Bob stores the public prekey and the signed prekey signature on the Signal server as part of his prekey bundle. By including the signed prekey signature in the prekey bundle, the Signal protocol ensures that users can authenticate each other's public prekeys, providing additional security and confidence in the integrity of their communication. Alice uses a digital signature verification algorithm to verify the signed prekey signature. By successfully verifying the prekey signature, Alice can trust the

prekey bundle is authentic and belongs to Bob, ensuring the integrity of the keys used in their secure communication.

(4) Alice creates a temporary public-private key pair specifically for this session. These keys are called ephemeral because they exist briefly and are discarded once the session is established. Generating an ephemeral key pair for each session ensures forward secrecy in the protocol.
(5) Alice computes a shared secret using her private ephemeral key and Bob's public prekey which she obtained from Bob's prekey bundle. Both Alice and Bob can independently calculate the same shared secret because the ECDH algorithm allows them to derive it using their respective private keys and the other's public key. A shared secret key is essential for establishing a secure communication channel between two parties.
(6) Alice initializes a double ratchet algorithm, securing end-to-end encryption with Bob using two ratchets: the DH and the symmetric-key ratchet. The root and symmetric encryption keys, used initially, derive the chain keys that generate message keys. As communication proceeds, new DH key pairs are exchanged, updating the root key and protecting against compromises. Each message uses a unique key, ensuring security even if individual message keys are compromised.
(7) Alice uses the double ratchet algorithm to encrypt her message. The encryption process involves generating a new message key, encrypting the message using the message key, and updating the symmetric-key ratchet state. The message is also tagged with a header containing the current ratchet state.
(8) Alice sends the encrypted message, along with the header, to Bob.
(9) Bob receives the encrypted message and the header. Using the header, he derives the same symmetric encryption key and root key that Alice used. He then initializes the double ratchet algorithm with these keys and decrypts the message using the appropriate message key.
(10) Upon successfully decrypting the message, Bob updates his double ratchet state to match Alice's.
(11) As Alice and Bob continue to exchange messages, the double ratchet algorithm ensures that new encryption keys are generated for each message, providing forward secrecy and preventing the compromise of one message key from affecting others.

5.8 Signal Security and Privacy Evaluation

Signal retains minimal user data, such as phone numbers, random keys, and the last connection time. It does not store message content, usernames, or contact lists on its servers. Signal does not maintain message logs or conversation history. Once messages are delivered and read, they are removed from Signal's servers. However, Signal stores the user's phone number, registration date, and last connection time. This information can provide a timeline for when a user was active on the platform.

5.9 Threema End-to-End Encryption

Threema employs e2ee for all forms of messages, ensuring user privacy. On setting up the application, it generates a unique pair of keys; the private key is stored on the user's device, while the public key is on Threema's server, linked to an eight-character ID. When messaging, Threema retrieves the recipient's public key using their Threema ID. Threema uses a "verification level" system for the identity of the person being messaged. The levels are red (least certain), orange (certain), green (most certain), and blue (for Threema Work internal contacts). To achieve the highest level of certainty (green), the sender can scan the QR code of the recipient's Threema ID and public key.

Threema uses a secure method called the "Box" model from the Networking and Cryptography library (NaCL) to encrypt messages. Following is a scenario of how message encryption takes place when Alice sends a message to Bob (See Figure 6):

(1) Threema generates a set of private and public keys for both Alice and Bob during installation. The private key is generated at random and securely installed in the device. It then creates a public key using Elliptic Curve (Curve25519) and sends it to the server.
(2) The server stores the public key and generates a random Threema ID consisting of 8 characters and sends it to the app. The app stores the Threema ID and the public and private keys securely in the user's device.
(3) Alice obtains Bob's public key from a secured channel.
(4) Alice uses ECDH with a specific curve (Curve25519) and combines Bob's public key and her private key. This creates a shared secret key that only Alice and Bob can access. Bob can access the shared secret due to the properties of the elliptic curve because the shared key remains the same if the keys are swapped.
(5) Alice generates a random Number Used Once (NOnce). A NOnce is a random value generated for authentication or encryption. NOnce uses diversity and prevents key reuse, which makes cryptographic systems secure.
(6) Threema encrypts her message using XSalsa20 cipher with the shared secret and NOnce.
(7) Then Message Authentication Code (MAC) using Poly1305 is added to ensure message authenticity. Alice sends the MAC, encrypted message, and NOnce to Bob.
(8) Bob Recreate Secure Storage Key (SSK) Using ECDH and CURVE25519.
(9) Bob Decrypts Message Using SSK and NOnce.
(10) Bob Verified Message Authenticity Using POLY1305 Mac

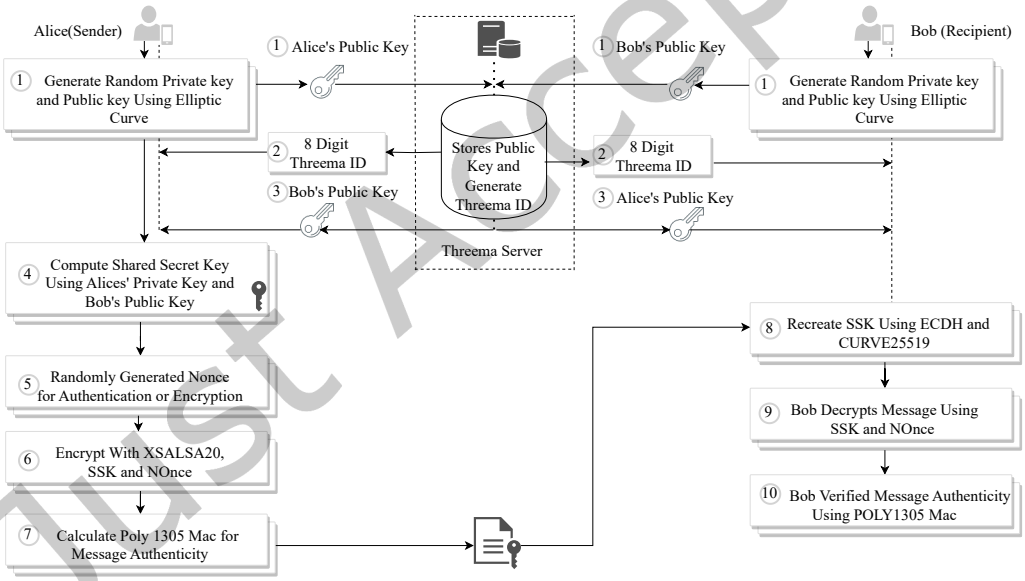

Fig. 6. Threema End-To-End Encryption

5.10 Threema Security and Privacy Evaluation

Threema's white paper emphasizes its commitment to user privacy and security, asserting that its data usage is limited to essential message delivery. Threema states it does not store metadata or generate log files, although specific data elements utilized are not explicitly detailed.

Threema allows for usage anonymity, with no need for users to divulge phone numbers, email addresses, or sync contacts. The anonymity of users is preserved as they can interact on the platform without any personally identifiable information disclosure. In instances where a phone number is provided, Threema employs a privacy-preserving technique where it stores only hashed values of these numbers on its servers, never the actual numbers. In case users opt to sync their contacts, Threema locally hashes the phone numbers on the device before transferring these hashed values to the servers. This process allows Threema to cross-reference these hashes with the stored ones to identify matches. The emphasis on privacy is further reinforced by Threema's policy of deleting any synced data from its servers post-synchronization, thereby bolstering user privacy and security.

KF 5. While encryption ensures user privacy, it significantly hinders forensic investigations. Robust protocols like the Signal Protocol make accessing message content nearly impossible without decryption keys. While encrypted message content is often inaccessible, metadata such as timestamps, sender and receiver identifiers, message sizes, and communication patterns can still be obtained and analyzed. This metadata provides valuable insights into user behavior and interactions, which can be critical in forensic investigation [30, 31, 124].

RG 4. Future digital forensics research of secure messaging applications should focus on enhancing data acquisition and analysis techniques to handle encrypted data, as current methods often struggle with robust encryption protocols.

5.11 Secure Messaging Protocol Analysis

To systematically analyze and evaluate secure messaging applications, we adopted a set of key parameters derived from the methodologies and frameworks established in the literature. Specifically, our evaluation criteria are based on the comprehensive analyses presented by Johansen et al. [63] and by Unger et al. [134]. These parameters encompass essential aspects of secure messaging protocols, including end-to-end encryption, confidentiality, integrity, authentication, perfect forward secrecy, future secrecy, deniability, synchronicity, multidevice support, key management and change handling, verification process, and additional security features. Table 3 presents our analysis.

6 Secure Messaging Applications Attack Surface and Vulnerabilities

Encryption and other security features of secure messaging applications can help protect user data and privacy, but if the application itself has security flaws, that privacy can still be violated. In cybersecurity, attack vectors are areas of exposure or weakness opening an application to an attack. The totality of an application's attack vectors is called the attack surface. This section considers the attack surface of WhatsApp, Telegram, Signal, Wickr, and Threema. These applications can also be exposed by vulnerabilities in third-party software or infrastructure (e.g., vulnerabilities on an Android device where the messenger is installed); these vulnerabilities are discussed when relevant.

6.1 CVEs and CVSS Scores

CVEs were created in 1999 to help track publicly exposed vulnerabilities in software packages. CVEs quickly became an industry standard and were adopted by the NIST in 2002 [88]. CVEs serve as a unique identifier for a vulnerability. They also contain extensive data about a vulnerability, including a detailed description of the issue, affected software packages, and historical data such as the vulnerability's publication date. CVEs can be in disputed status, meaning the vendor or a third party does not agree the issue is a vulnerability. CVEs are also associated with a score. Scores are determined using the Common Vulnerability Scoring System (CVSS). CVSS

Table 3. Evaluation of Secure Messaging Applications.✓Fully supports the parameter, ●Partially supports the parameter, ✗Does not support the parameter.

Parameter	Definition and Impact	WhatsApp	Wickr	Signal	Threema	Telegram
Confidentiality	Ensures message content remains secret and is accessible only to intended recipients. Protects sensitive information from unauthorized access.	✓	✓	✓	✓	●
Integrity	Guarantees message content has not been altered during transmission. Prevents tampering and ensures message authenticity.	✓	✓	✓	✓	✓
Authentication	Confirms the identity of communicating parties. Protects against impersonation attacks.	✓	✓	✓	✓	●
PFS	Ensures compromise of long-term keys does not compromise past session keys. Protects past communications from future key compromises.	✓	✓	✓	✓	●
Future Secrecy	Ensures compromise of long-term keys does not allow decryption of future messages. Maintains ongoing security of communications.	✓	✓	✓	✓	●
Deniability	Ensures messages cannot be proven to have been sent by a specific user after a conversation. Protects users from incrimination based on past messages.	✓	✓	✓	✓	●
Synchronicity	Whether messaging requires both parties to be online simultaneously (synchronous) or allows offline message delivery (asynchronous). Asynchronous messaging enhances usability.	●	✓	✓	✓	✓
Multidevice Support	Ability for users to access messages across multiple devices. Increases usability and ensures a seamless experience.	●	●	✓	●	✓
Key Management and Change Handling	How the application handles cryptographic key changes, including notifications and re-encryption of messages. Ensures security and prevents undetected attacks.	✓	✓	✓	✓	●
Verification Process	Methods for users to verify the identity of their contacts. Enhances security by confirming contact identities and reducing impersonation risk.	✓	✓	✓	✓	●
Additional Security Features	Extra measures such as passphrase protection, two-step verification, screen security, and device management. Provides additional layers of security and user confidence.	●	✓	✓	✓	●

was developed in 2004 and was adopted by NIST in 2007. It provides a standardized, publicly available method to score the severity of an issue and its ability to be exploited. The scores are associated with 4 severity levels: Low, Medium, High, and Critical. CVSS has had 3 major versions. In this paper, we focus on the most recent, version 3.1. Scores from the previous version are included when there is no version 3.1 information (i.e., when a vulnerability was published before the release of CVSS 3.1)[4].

6.2 WhatsApp Vulnerabilities

WhatsApp has had 47 CVEs logged since May 2015. They have an average score of 5.88 (6.4 median) on the V2 scale, with numbers ranging from 2.1 to 10 and an average score of 7.79 (7.8 median) on the V3 scale, with

[4]CVSS v2 was retired July 13th, 2022: https://nvd.nist.gov/general/news/retire-cvss-v2

numbers ranging from 3.3 to 10. More than half of WhatsApp's vulnerabilities have been major in scope, and half of those have been considered Critical. Overflow issues are common in the WhatsApp codebase; stack, heap, and buffer overflows make up 11 of the major issues, with 6 of them Critical. Other major vulnerabilities include heap corruption, URL validation, URI spoofing, cross-site scripting, and cache configuration issues. The most recently published CVEs for WhatsApp were in September of 2022. Both dealt with remote code execution on video calls. WhatsApp claimed both vulnerabilities were fixed in version 2.22.16.12. Older CVEs all reference a final version affected, indicating the vulnerabilities have been patched.

Vulnerabilities in the supplemental software were frequent near the beginning of WhatsApp. A vulnerability in the Android implementation of AES made it possible for attackers to decrypt a WhatsApp database file [111]. This vulnerability was still exploitable in January 2023, when Bogos et al. [27] were able to use it and a UFED to obtain a database file and its decryption key. The use of network sniffing software initially could intercept the cell phone number of a user, which one researcher used in a profiling attack [75].

6.3 Telegram Vulnerabilities

Telegram has recorded 36 CVEs since December 2016, averaging 4.41 (4.3 median) on the V2 scale (range: 2.1–7.5) and 6.36 (6.1 median) on the V3 scale (range: 2.4–9.8). Eleven are major, with six in disputed status, including two major ones. Key issues include directory traversal, improper sanitation, and plaintext transmission of credentials or secret chats. The CVEs, published in December 2022, include a disputed Medium cross-site scripting issue in Telegram Desktop and a High SQL injection in a WordPress plugin, which was patched. Older CVEs indicate patched vulnerabilities.

Telegram has been criticized for its proprietary encryption protocol, often considered less secure than public algorithms. In 2016, researchers exploited a replay attack in MTProto [128]; Telegram acknowledged the issue but deemed it limited in scope. It was patched in January 2017. In 2019, a vulnerability allowed attackers to embed malicious code in animated stickers, leading to data theft, malware installation, or device takeover [1]. In late 2020, an Iranian hacker group exploited an Android backdoor to bypass Telegram's two-factor authentication, targeting Iranian users [92].

6.4 Signal Vulnerabilities

Signal has had 11 CVEs logged since April 2018. They have an average score of 5.33 (5.0 median) on the V2 scale, with numbers ranging from 1.9 to 7.8, and an average score of 6.82 (7.3 median) on the V3 scale, with numbers ranging from 3.3 to 9.8. The most recent CVEs were in January of 2023. Both issues dealt with attachments in the desktop version of Signal, and they are both in disputed status. Older CVEs all reference a final version affected, indicating the vulnerability has been patched.

Much of the research about the Signal messenger application indicates it is one of the safer options, but these papers start with the assumption a user is dedicated to one device. Wichelmann et al. utilized Signal's implementation of the Sesame protocol to establish multiple devices for a single user. They were able to show how a user's communications can be compromised by registering a malicious device.

6.5 Threema Vulnerabilities

Threema is only associated with 1 CVE, CVE-2023-22899. The vulnerability is in a third-party package, Zip4j, and involves the decryption process of a zip file. The CVE is classified as Medium severity with a score of 5.9. Paterson et al. were able to exploit this vulnerability to create backup zip files. In the same paper, these researchers present 7 total attacks on Threema: a key compromise, a vouch box forgery, message reordering and deletion, message replay, kompromat, cloning using the Threema ID, and a compression side-channel attack. They present these issues as violations of security standards and consider them design weaknesses.

6.6 Wickr Vulnerabilities

Wickr was the only application analyzed with zero identified CVEs. This may be related to the application's size. Wickr has one of the smallest user bases of the secure messenger applications, and according to Watanabe et al. [139], messengers with bigger user bases tend to have more vulnerabilities than smaller applications. Less people using a secure messenger may mean less people are testing for exploits. It is also possible Wickr is more secure than other messengers. Walnycky et al. [137] conducted forensic analysis on 20 social messenger applications for Android devices and were unable to find any security vulnerabilities or uncover user information on Wickr.

> **RG 5.** The increasing use of multiple devices for a single user in secure messaging applications introduces new security challenges. Currently, no comprehensive research has been conducted on secure device registration practices, synchronization methods, and how to exploit multi-device usage in secure messaging applications.

7 Analysis

In our comprehensive review, we identified a total of 35 technical papers focusing on forensic analysis of secure messaging applications.

7.1 Forensic Analysis Systematization

We systematically categorized the various aspects of forensic investigations, including analysis types, artifacts retrieved, methodologies, tools used, and challenges encountered. This comprehensive overview is critical for understanding the current landscape and identifying areas for future research.

For each application, we listed the specific artifacts retrieved, their storage locations, the tools employed for forensic analysis, and the open problems and challenges unique to each application. The tables provided summarize the key findings from our review of the literature, offering a clear and concise reference for researchers and practitioners in the field.

> **KF 6.** Volatile memory (RAM) analysis can be a valuable technique for recovering volatile data that is not stored permanently on the device. Analyzing RAM can provide critical insights and artifacts, such as decrypted data, that are otherwise inaccessible in encrypted storage.

Table 4. The table categorizes various aspects of forensic investigations into WhatsApp, Signal, Telegram, Wickr, Threema including the types of analysis conducted, the artifacts retrieved, the methodologies employed, the tools used, and the challenges encountered. The symbols represent different levels of presence or compliance for each feature: ● indicates a strong presence ; ◗ denotes partial presence; and ❍ signifies an absent feature. Additionally, checkmarks (✓) indicate the presence of a specific type of analysis, tool, or challenge, while crosses (×) indicate their absence. (■) indicates the presence of Static and Dynamic Method, on the contrary (❑) indicates their absence. The table is organized by platform, year, and reference, providing a comprehensive overview of the current state of forensic analysis research on Secure Messaging Application. (*) highlights the research regarding the proposed tools for forensic analysis.

Platform	Year	Ref	Analysis			Artifacts Retrieved							Method		Tools		Challenges				
			Network	Device	Memory	Databases	Multimedia	Logs	Preferences	Account Info	Message Content	Network	Static	Dynamic	Forensic	Analysis	Encryption/Decryption	Access/Privileges	Traffic Analysis	Version Dependency	Interception/Manipulation
WhatsApp	2021	[6]	✓	×	×	●	❍	◗	❍	●	◗	●	■	■	×	✓	✓	✓	×	×	✓
	2021	[89]	×	✓	×	●	●	●	●	●	●	❍	■	❑	✓	✓	✓	✓	×	✓	✓
	2021	[144]	✓	✓	×	❍	●	●	❍	●	●	●	■	■	✓	✓	✓	×	✓	×	✓
	2021	[136]	✓	×	×	●	❍	❍	❍	❍	●	❍	■	■	×	✓	×	✓	×	✓	✓
	2020*	[13]	×	×	✓	❍	❍	❍	❍	❍	●	❍	■	❑	✓	×	×	✓	×	✓	×
	2020	[86]	×	✓	×	❍	●	❍	❍	❍	❍	●	■	❑	✓	✓	×	✓	×	×	✓
	2020	[112]	✓	×	✓	●	●	●	❍	●	●	❍	■	❑	✓	✓	×	✓	×	×	×
	2020	[95]	✓	✓	×	●	❍	●	❍	●	●	●	■	❑	✓	✓	×	✓	✓	✓	✓
	2019	[122]	✓	×	×	❍	❍	❍	❍	❍	❍	●	❑	■	✓	✓	✓	✓	✓	×	✓
	2018	[93]	×	✓	×	●	●	●	●	●	●	❍	■	❑	✓	✓	✓	✓	×	✓	×
	2015*	[67]	×	×	✓	❍	❍	❍	❍	❍	❍	●	❑	■	✓	✓	✓	✓	✓	✓	✓
	2015	[120]	×	✓	×	●	●	❍	❍	●	●	❍	■	❑	✓	✓	✓	✓	×	✓	✓
	2013	[131]	✓	×	×	●	●	●	❍	●	●	❍	■	❑	✓	✓	✓	✓	×	✓	✓
Signal	2022	[123]	×	✓	✓	●	●	●	●	❍	●	❍	■	■	✓	✓	✓	✓	✓	✓	✓
	2022	[147]	✓	✓	✓	✓	●	●	●	●	●	❍	■	■	×	✓	✓	✓	×	✓	×
	2022	[106]	×	✓	×	❍	◗	❍	❍	●	❍	❍	■	❑	✓	×	✓	✓	×	×	×
	2019	[66]	×	✓	✓	●	●	❍	●	●	●	❍	■	■	✓	×	✓	✓	×	✓	×
	2021	[5]	✓	×	×	❍	❍	❍	❍	❍	❍	●	■	❑	×	✓	✓	×	✓	×	×
	2021	[72]	×	✓	×	●	●	●	●	●	●	❍	■	■	✓	×	✓	✓	×	×	×
	2016	[117]	✓	×	×	❍	❍	❍	❍	❍	❍	●	■	■	×	✓	✓	✓	✓	×	✓
Telegram	2022	[46]	×	×	✓	◗	◗	❍	❍	●	●	❍	■	■	✓	×	✓	✓	×	✓	✓
	2022	[11]	✓	×	×	❍	❍	❍	❍	❍	●	●	■	■	×	×	✓	✓	✓	✓	✓
	2022	[105]	×	✓	×	❍	●	●	●	●	●	❍	■	■	×	✓	✓	✓	✓	✓	✓
	2022	[91]	×	✓	×	●	●	●	●	●	●	❍	■	■	✓	✓	✓	✓	×	✓	✓
	2020	[141]	×	✓	×	●	●	●	❍	●	●	❍	■	■	✓	✓	×	✓	×	✓	✓
	2018	[53]	✓	✓	×	●	●	●	●	●	●	❍	■	■	✓	✓	✓	✓	×	✓	✓
	2018	[54]	×	✓	×	●	●	●	●	●	●	❍	■	■	✓	✓	✓	✓	×	✓	✓
	2018	[54]	×	✓	×	●	●	●	●	●	●	❍	■	■	✓	✓	✓	✓	×	✓	✓
	2016	[114]	×	✓	×	●	●	●	●	●	●	❍	■	■	✓	✓	✓	✓	✓	✓	✓
Wickr	2022	[123]	×	✓	×	●	●	●	●	●	●	❍	■	■	✓	✓	✓	✓	×	✓	✓
	2021	[68]	×	✓	✓	●	●	❍	◗	●	●	❍	■	■	✓	✓	✓	✓	×	✓	×
	2016	[23]	×	✓	✓	●	●	❍	❍	●	●	❍	■	■	✓	✓	✓	✓	×	✓	×

Continued on next page

Table 4 – continued from previous page

Platform	Year	Ref	Analysis			Artifacts Retrieved							Method		Tools		Challenges				
			Network	Device	Memory	Databases	Multimedia	Logs	Preferences	Account Info	Message Content	Network	Static	Dynamic	Forensic	Analysis	Encryption/Decryption	Access/Privileges	Traffic Analysis	Version Dependency	Interception/Manipulation
	2013	[87]	×	✓	✓	❍	❍	❍	❍	❍	❍	❍	■	■	×	✓	✓	✓	×	✓	×
	2022	[123]	×	✓	✓	●	●	❍	●	❍	●	❍	■	■	✓	✓	✓	✓	×	×	×
Threema	2023	[19]	×	✓	×	●	◗	●	◗	●	●	◗	■	■	✓	✓	✓	✓	✓	✓	✓

RG 6. There is a need for standardized forensic methodologies to ensure consistent data extraction and analysis across different secure messaging platforms and operating systems. The current lack of cross-platform forensic analysis studies leaves a significant gap in consistent methodologies.

RG 7. Comprehensive memory analysis studies focusing on Secure Messaging applications are lacking. Future research should explore memory analysis to capture ephemeral data that is not permanently stored.

RG 8. Developing methods to bypass or decrypt network-level encryption in secure messaging apps remains a critical challenge.

Table 5. Comprehensive Overview of Forensic Artifacts Locations for Secure Messaging Applications. This Table Provides a Detailed Breakdown of Where Various Forensic Artifacts Can Be Found Within Different Secure Messaging Applications, Based on Forensic Analysis From Multiple Studies. The Artifacts Are Categorized by Databases, Multimedia, Logs, Preferences, Account Information, Message Content, and Network Artifact Content. Each Entry Includes the Specific Locations and References to the Corresponding Research Papers.

Platform	Artifacts	Location
WhatsApp	Database	On Android main databases msgstore.db and wa.db are located in /data/data/com.whatsapp/databases/, containing user information, contacts, and messages [131]. In WhatsApp web, IndexedDB storage files are located in the browser's persistent storage [131].
	Multimedia	On Android, files are stored in /sdcard/WhatsApp/Media/ and within the app's data folder [131].
	Logs	On Android, log files are stored in the "/data/data/com. whatsapp/files/Logs" directory, and on iOS devices in application's sandbox directory. [124].
	Account Information	Account details, including contact lists and personal information, are stored in the wa.db database [131].
	Message Content	Plain text messages are stored in the msgstore.db database. Deleted messages can also be found in RAM during live data forensics [131].
	Network Artifact Content	Network traffic analysis using tools like Wireshark can provide metadata about the communication, including IP addresses and port numbers used in WhatsApp calls [95, 122].

Table 5 – continued from previous page

Platform	Artifacts	Location
Signal	Database	Signal's main database signal.db is located in /data/data/org.thoughtcrime.securesms/databases/. This file stores user information, contacts, and messages [123].
	Multimedia	Multimedia files (images, videos, audio) are stored in /storage/emulated/0/Signal/ and within the app_parts directory in the application's data folder [66, 123].
	Logs	Encrypted logs are found in the cache/log directory within the application's data folder, providing details about user activity and app usage.
	Preferences & Account Information	User preferences and account information are stored in shared preference files such as SecureSMS-Preferences.xml and org.thoughtcrime.securesms_preferences.xml. These files contain data on user settings, account configurations, and application behavior [123].
	Message Content	Plain text messages are stored in the signal.db database, encrypted using SQLCipher [123].
	Network Artifact Content	Network traffic analysis using Wireshark can provide metadata about the communication, although the message content remains encrypted [123].
Telegram	Database	On Android, data is dispersed across /data/data/org.telegram.messenger and /data/media/0/Telegram [28, 114]. These locations contain databases like cache4.db (SQLite), which stores critical artifacts such as user information, contacts, messages, and logs.
	Message	On macOS, core application data is stored in /Applications/Telegram.app, while user data such as encrypted messages are located in /Users/$user/Library/Application Support/Telegram Desktop/tdata [53]. For Windows Phone, text messages are stored in the dialogs.dat file [54].
	Preferences	Extracted shared preferences stored in shared_prefs directory, including user configuration and settings [105].
Wickr	Database	On Android, the database file (wickr_db) is encrypted using SQLCipher, and the decryption passphrase is stored in the sk.wic file, encrypted using AES256-GCM and the Scrypt key derivation function (KDF). On iOS, Wickr does not encrypt the database file itself but encrypts the contents using AES256-GCM, with the decryption key stored in the ZPT column of the ZSECEX_ACCOUNT table [68]
	Multimedia	Files are renamed to GUID format and encrypted using AES256-GCM, with the decryption key stored in the messagePayload (Android) or ZBODY (iOS) column of the database.
	Preferences	Files are in the internal files directory with .prefs extensions. Decryption requires extracting keys from files like kcd.wic and kck.wic, and using AES256-GCM [123].
Threema	Database	The encrypted database files, such as threema4.db, are located in the INTERNAL/databases/ directory, which also contains another unencrypted database, threema-nonce-blob4.db. These files can be seen without rooting the device [123].
	Preference	Preference files (ch.threema.app_preferences.xml, crs-pref_message_drafts, crs-pref_push_token, crs-pref_threema_safe_masterkey), are stored within the INTERNAL/files/ and INTERNAL/shared_prefs/ directories [123].

Table 5 – continued from previous page

Platform	Artifacts	Location
	Multimedia	Encrypted images and videos are stored in the EXTERNAL/Android/data/ch.threema.app/files/data/ directory, with thumbnails denoted by a _T suffix in their filenames. Cached image data can be found in the INTERNAL/cache/image_manager_disk_cache/directory. When a backup is created using the app, it is saved as a zip file in the EXTERNAL/Threema/Backups/ directory. An 67-character decryption key can be obtained from file key.dat and used to decrypt all this information [123].

RG 9. Volatile memory analysis should be explored because RAM may contain decrypted versions of data that are otherwise inaccessible in encrypted storage, providing critical insights.

Table 6. Overview of Forensic Tools and Their Effectiveness in Secure Messaging Applications. ✓indicates compatibility with the application; Memory and Network columns denote applicability in forensic investigations.

Tool	Signal	Telegram	Wickr	Threema	WhatsApp	Artifacts		Forensic Methodology	Memory	Network
						Type	Retrieved			
Wireshark [5]	✓	✓	✗	✗	✓	Network Traffic	Packet Metadata	Traffic Pattern Inspection	✗	✓
Frida [123]	✓	✗	✗	✓	✓	Runtime Artifacts	Decrypted Logs, Multimedia Files	In-memory Data Extraction	✓	✗
Signal Backup Decryptor	✓	✗	✗	✗	✗	Backup Data	Decrypted Backup Files	Full Backup Decoding	✓	✗
MOBILedit [106]	✓	✗	✗	✗	✓	Application Metadata	Contact Info, App Data	Data Recovery from Storage	✓	✗
BEC [91]	✗	✓	✗	✗	✓	Database	Messages, Multimedia Files	SQL-based Data Analysis	✓	✓
Magnet AXIOM [91]	✗	✓	✗	✗	✓	Database, Logs	cache4.db, Preferences, msgstore.db	File System and Log Analysis	✓	✓
Checkra1n/Magisk [91]	✗	✓	✗	✗	✓	System Data	Rooted Device Data, Logs	Bypassing Security Restrictions	✓	✓
IDA Pro [123]	✗	✗	✓	✗	✗	Binary Code	Disassembled Binaries	Static Code Inspection	✓	✗
JEB Decompiler [123]	✗	✗	✓	✓	✗	App Code	Decompiled Encryption Logic	Algorithmic Flow Analysis	✓	✗
Android Debug Bridge (ADB) [123]	✗	✗	✗	✓	✓	Storage Data	Databases, Preferences	Extracting App-Level Storage	✓	✗
UFED [131]	✗	✗	✗	✗	✓	Full Device Extraction	msgstore.db, wa.db, Contacts	Logical	✓	✗
WhatsApp Xtract [120]	✗	✗	✗	✗	✓	Database, Logs	Decrypted WhatsApp Data	Forensic Analysis of Encrypted Databases	✓	✗
Autopsy [131]	✗	✗	✗	✗	✓	Storage Artifacts	WhatsApp Deleted Data Recovery	Data Carving	✓	✗

KF 7. Frequent updates of SM applications complicate consistent forensic investigations [105, 141]. To address this issue, forensic tools should incorporate version-specific modules that can dynamically adapt to different database schema. Additionally, creating a collaborative and open-source database of schema changes can help forensic analysts stay updated with the latest modifications across various secure messaging applications.

RG 10. Future studies should focus on creating user-friendly forensic tools that respect privacy and automatically collect information post-security breaches without invading privacy. This includes smart forensic tools that can adapt to different secure messaging applications and provide reliable data without requiring extensive technical expertise.

8 Legal and Ethical Dimensions of Secure Messaging Forensics

Forensic investigations must follow a complex legal landscape, particularly regarding data privacy laws such as the General Data Protection Regulation (GDPR) and California Consumer Privacy Act (CCPA). These regulations enforce strict limitations on how investigators can access, store, and analyze user data, often requiring warrants before digital evidence can be lawfully obtained [22]. The challenges increase when dealing with cloud-stored data, as multi-jurisdictional legal frameworks complicate access to evidence across international borders [50]. Additionally, ensuring forensic soundness while complying with legal requirements requires careful handling of cross-border investigations, particularly in securing authorization before retrieving data from cloud service providers. Without proper legal guidelines, forensic findings risk being considered inadmissible in court [57]. To mitigate these concerns, chain-of-custody procedures become crucial. Every step in handling, duplicating, or decrypting digital evidence should be transparently documented, ensuring that the data remain authentic and unmodified [39].

Beyond legal constraints, ethical concerns pose additional challenges in forensic investigations. The practice of bypassing encryption to access secured data raises privacy rights concerns, as it often involves exploiting vulnerabilities in applications or leveraging advanced forensic techniques like RAM acquisition and cold boot attacks [63]. The exploitation of vulnerabilities could potentially weaken overall cybersecurity, making systems more susceptible to malicious attacks [112].

KF 8. The need to thoroughly investigate digital evidence often contradicts with the responsibility to uphold user privacy. Balancing forensic acquisition and strict data protection requirements remains a challenging task for investigators.

9 Discussions and Recommendations

`Emerging Secure Messengers:` Apart from the secure messaging applications examined in this study, other platforms—such as Element (Matrix), Session, and Briar—also implement end-to-end encryption. However, to the best of our knowledge, only one prior study has addressed the user experience of Briar [2], and no forensic analyses have been conducted on any of these applications. Due to their relatively small user bases compared to mainstream messaging services, these platforms remain understudied in forensic contexts, representing a clear opportunity for future research.

RG 11. Element (Matrix), Session, and Briar are increasing in usage, there is a noticeable lack of comprehensive forensic acquisition and analysis methodologies for these applications, indicating a significant gap for future research.

`AI-Driven Forensic:` Recent advances in AI are reshaping digital forensics in secure messaging. Investigators can employ AI-based models to automate data classification, detect anomalies in large volumes of seized data, and reconstruct partial conversation histories even when message content is encrypted or has self-destructed [89, 112]. For instance, real-time analytics of suspicious user activity or device logs could alert forensic teams to patterns of ephemeral communication that warrant deeper examination [123]. Additionally, pattern-recognition algorithms may help cluster related artifacts across multiple apps or platforms, revealing user behavior that would be labor-intensive to uncover manually [63].

`Quantum Computing and the Future of Encryption:` E2EE is strong against cracking attacks from classical computers, but quantum computing may weaken common encryption methods (e.g., RSA, Diffie-Hellman, elliptical curves) [145]. Using quantum algorithms such as Shor's algorithm, a quantum machine could tractably factor large numbers to decrypt messages, threatening user privacy [12]. For digital forensics, the move toward post-quantum cryptography would make it harder to break encryption, meaning investigators might need to rely on other clues such as metadata or logs. At the same time, if law enforcement gains quantum tools before messaging apps upgrade to quantum-safe protocols, they could exploit outdated systems to decrypt data. This shows how the advent of quantum computing can motivate the strengthening of future encryption to improve user privacy and create new opportunities for forensic access if some services fall behind in upgrades [94].
`Recommendations:` Several future recommendations can be made based on our analysis. First, apart from Signal and Wickr, current secure messaging applications rely on centralized servers, which may become targets for attacks, censorship, or legal pressures. Research should focus on creating decentralized peer-to-peer messaging architectures that can provide greater resilience and autonomy [18]. Second, research should focus on improving user-friendly design in secure messaging applications. For instance, Telegram implements E2EE only in its Secret Chat feature. The design of the application does not adequately inform new users about this, potentially leading to misconceptions about Telegram's security level. Third, the forensic community should focus on creating smart forensic tools that automatically collect information after a security breach without invading people's privacy. This focus on easy-to-use and privacy-aware tools could greatly improve the effectiveness and adoption of these DF tools, especially in the domain of secure messaging applications [90]. Fourth, inclusion of forward secrecy and deniability in secure messaging applications, although privacy-boosting, complicates digital forensics by protecting and potentially hiding messages. Future research needs to develop new methods and tools to work effectively with these platforms and address the challenges posed by forward secrecy and deniability. Finally, the forensic research is conducted worldwide by various institutions, producing valuable information published in academic journals and conferences. The creation and sharing of open-source forensic tools is a significant advancement, yet their technical complexity limits the general public use. As a result, user-friendly DF solutions are needed that allow widespread adoption without extensive technical knowledge.

10 Conclusions

In this paper, we explored the ever-changing landscape of secure messaging applications from a forensic perspective. Our study offered a comprehensive review, spanning from the inception of these platforms to the present challenges they face in forensic examinations. With the goal of better understanding the basic principles that make these applications secure, we examined how encryption works, especially the process of end-to-end encryption. Using a diagrammatic representation, we explained the encryption process from sender to receiver to make these essential security measures easier to understand.

A key revelation of our research highlighted the contrast between traditional applications, such as WhatsApp, and forensic-resistant applications such as Wickr, Threema, and Signal. It became apparent that extracting evidence from the latter class of applications posed a unique set of challenges. By diving into the encryption systems, forensic analysis capabilities, data policies, evolution, and potential vulnerabilities of each application, we believe that our research served as a valuable roadmap for both users and investigators navigating the space of secure messaging applications. In summary, this paper thoroughly scrutinized secure messaging applications through the lens of forensic analysis. We anticipate that our work will act not only as an informative guide to the present state of secure messaging applications but also lay the groundwork for future research in this rapidly evolving domain. This continuous change, defined by an ongoing cycle of problems and solutions, highlights both the complexity and the necessity of our work. As we continue to navigate this increasingly intricate landscape, let this work serve as a guide toward optimal privacy and data protection.

References

[1] 2021. *Shielder - Hunting for bugs in Telegram's animated stickers remote attack surface.* https://www.shielder.com/blog/2021/02/hunting-for-bugs-in-telegrams-animated-stickers-remote-attack-surface/

[2] Blerton Abazi and Renata Gegaj. 2022. A Comparative Study on the User Experience on Using Secure Messaging Tools. In *Big Data Privacy and Security in Smart Cities*. Springer, 119–131.

[3] Bery Actoriano and Imam Riadi. 2018. Forensic Investigation on WhatsApp Web Using Framework Integrated Digital Forensic Investigation Framework Version 2. *International Journal of Cyber-Security and Digital Forensics (IJCSDF)* 7, 4 (2018), 410–419.

[4] Somaya Adwan and Fahad Salamah. 2018. A manual mobile phone forensic approach towards the analysis of Whatsapp seven-minute delete feature. In *2018 21st Saudi Computer Society National Computer Conference (NCC)*. IEEE, 1–5.

[5] Asmara Afzal, Mehdi Hussain, Shahzad Saleem, M Khuram Shahzad, Anthony TS Ho, and Ki-Hyun Jung. 2021. Encrypted network traffic analysis of secure instant messaging application: A case study of signal messenger app. *Applied Sciences* 11, 17 (2021), 7789.

[6] Waqas Ahmed, Faisal Shahzad, Abdul Rehman Javed, Farkhund Iqbal, and Liaqat Ali. 2021. Whatsapp network forensics: Discovering the ip addresses of suspects. In *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 1–7.

[7] Arafat Al-Dhaqm, Shukor Abd Razak, Richard Adeyemi Ikuesan, Victor R. Kebande, and Kamran Siddique. 2020. A Review of Mobile Forensic Investigation Process Models. *IEEE Access* 8 (2020), 173359–173375. doi:10.1109/access.2020.3014615

[8] Abdalazim Abdallah Mohammed Alamin and A Mustafa. 2015. A survey on mobile forensic for android smartphones. *IOSR Journal of Computer Engineering (IOSR-JCE)* 17, 2 (2015), 15–19.

[9] Mashari Alatawi and Nitesh Saxena. 2023. Sok: An analysis of end-to-end encryption and authentication ceremonies in secure messaging systems. In *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 187–201.

[10] Martin R Albrecht, Lenka Mareková, Kenneth G Paterson, and Igors Stepanovs. 2021. Security Analysis of Telegram (Symmetric Part). https://mtpsym.github.io/. [Online; accessed 2023-01-19].

[11] Martin R Albrecht, Lenka Mareková, Kenneth G Paterson, and Igors Stepanovs. 2022. Four attacks and a proof for Telegram. In *43rd IEEE Symposium on Security and Privacy (IEEE S&P 2022)*.

[12] Aminah Albuainain, Jana Alansari, Samiyah Alrashidi, Wasmiyah Alqahtani, Jana Alshaya, and Naya Nagy. 2022. Experimental Implementation of Shor's Quantum Algorithm to Break RSA. In *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)*. IEEE, 748–752.

[13] Aisha Ali-Gombe, Alexandra Tambaoan, Angela Gurfolino, and Golden G Richard III. 2020. App-agnostic post-execution semantic analysis of Android in-memory forensics artifacts. In *Proceedings of the 36th Annual Computer Security Applications Conference*. 28–41.

[14] Khalid Alissa, Norah A Almubairik, Lamyaa Alsaleem, Deema Alotaibi, Malak Aldakheel, Sarah Alqhtani, Nazar Saqib, Samiha Brahimi, and Mubarak Alshahrani. 2019. A comparative study of WhatsApp forensics tools. *SN Applied Sciences* 1, 11 (2019), 1–10.

[15] Joël Alwen, Binyi Chen, Krzysztof Pietrzak, Leonid Reyzin, and Stefano Tessaro. 2017. Scrypt is maximally memory-hard. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 33–62.

[16] Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. 2019. The double ratchet: security notions, proofs, and modularization for the signal protocol. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 129–158.

[17] Stefania Andries, Andrei-Daniel Miron, Andrei Cristian, and Emil Simion. 2022. A survey on the security protocols employed by mobile messaging applications. *Cryptology ePrint Archive* (2022).

[18] Henry Ang, Shaohui Guo, and Jingyi Bian. [n. d.]. Implementing A Decentralized Messaging Application. ([n. d.]).

[19] Cosimo Anglano, Massimo Canonico, Andrea Cepollina, Davide Freggiaro, Alderico Gallo, and Marco Guazzone. 2023. Enabling the forensic study of application-level encrypted data in Android via a Frida-based decryption framework. In *Proceedings of the 18th International Conference on Availability, Reliability and Security*. 1–10.

[20] H Azhar, Rhys Cox, and Aimee Chamberlain. 2020. Forensic investigations of popular ephemeral messaging applications on android and ios platforms. *International Journal on Advances in Security* 13, 1 & 2 (2020), 41–53.
[21] Ibrahim (Abe) Baggili. 2014. Do not share your location with your friends on WhatsApp until this issue is fixed! https://www.linkedin.com/pulse/20140414163047-105475546-do-not-share-your-location-with-your-friends-on-whatsapp-until-this-issue-is-fixed/
[22] Konstantia Barmpatsalou, Tiago Cruz, Edmundo Monteiro, and Paulo Simoes. 2018. Current and future trends in mobile device forensics: A survey. *ACM Computing Surveys (CSUR)* 51, 3 (2018), 1–31.
[23] T Barton and MHB Azhar. 2016. Forensic analysis of the recovery of Wickr's ephemeral data on Android platforms. In *The First International Conference on Cyber-Technologies and Cyber-Systems, IARIA*. 35–40.
[24] Daniel J Bernstein. 2005. The Poly1305-AES message-authentication code. In *International workshop on fast software encryption*. Springer, 32–49.
[25] Daniel J Bernstein. 2009. Cryptography in nacl. *Networking and Cryptography library* 3, 385 (2009), 62.
[26] Prabir Bhattacharya, Mourad Debbabi, and Hadi Otrok. 2005. Improving the Diffie-Hellman secure key exchange. In *2005 International Conference on Wireless Networks, Communications and Mobile Computing*, Vol. 1. IEEE, 193–197.
[27] Corina-Elena Bogos, Răzvan Mocanu, and Emil Simion. 2023. A security analysis comparison between Signal, WhatsApp and Telegram. https://eprint.iacr.org/undefined/undefined
[28] Alexander I Borodin, Roman R Veynberg, Dmitry V Pisarev, and Oleg V Litvishko. 2019. Simulation of artefact detection in Viber and Telegram instant messengers in Windows operating systems. *Business Informatics* 13, 4 (eng) (2019), 39–48.
[29] Triawan Adi Cahyanto, M Ainul Rizal, Ari Eko Wardoyo, Taufiq Timur Warisaji, et al. 2022. Live Forensic to Identify the Digital Evidence on the Desktop-based WhatsApp. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)* 6, 2 (2022), 213–219.
[30] Harlan Carvey. 2014. *Windows forensic analysis toolkit: advanced analysis techniques for Windows 8*. Elsevier.
[31] Eoghan Casey. 2011. *Digital evidence and computer crime: Forensic science, computers, and the internet*. Academic press.
[32] Cellebrite. 2020. Helping Law Enforcement Lawfully Access The Signal App. https://cellebrite.com/en/cellebrites-new-solution-for-decrypting-the-signal-app/
[33] Jiwon Choe, Tali Moreshet, R Iris Bahar, and Maurice Herlihy. 2019. Attacking memory-hard scrypt with near-data-processing. In *Proceedings of the International Symposium on Memory Systems*. 33–37.
[34] Grayson Clary. 2016. The Flaw in ISIS's Favorite Messaging App. https://www.theatlantic.com/technology/archive/2016/01/isiss-favorite-messaging-app-has-a-security-problem/422460/. [Online; accessed 2023-01-19].
[35] Jordan Crook. 2014. Hole In WhatsApp For Android Lets Hackers Steal Your Conversations. https://techcrunch.com/2014/03/12/hole-in-whatsapp-for-android-lets-hackers-steal-your-conversations/. [Online; accessed 2023-01-26].
[36] David Curry. 2023. Signal Revenue & Usage Statistics (2023). https://www.businessofapps.com/data/signal-statistics/
[37] Alison Deutsch. 2022. WhatsApp: The Best Meta Purchase Ever? https://www.investopedia.com/articles/investing/032515/whatsapp-best-facebook-purchase-ever.asp.
[38] D Eastlake 3rd and T Hansen. 2011. RFC 6234: US secure hash algorithms (SHA and SHA-based HMAC and HKDF).
[39] Hany M Elgohary, Saad M Darwish, and Saleh Mesbah Elkaffas. 2022. Improving uncertainty in chain of custody for image forensics investigation applications. *IEEE Access* 10 (2022), 14669–14679.
[40] Robert E Endeley et al. 2018. End-to-end encryption in messaging services and national security—case of WhatsApp messenger. *Journal of Information Security* 9, 01 (2018), 95.
[41] Ksenia Ermoshina, Francesca Musiani, and Harry Halpin. 2016. End-to-end encrypted messaging protocols: An overview. In *International Conference on Internet Science*. Springer, 244–254.
[42] Levent Ertaul, Manpreet Kaur, and Venkata Arun Kumar R Gudise. 2016. Implementation and performance analysis of pbkdf2, bcrypt, scrypt algorithms. In *Proceedings of the international conference on wireless networks (ICWN)*. The Steering Committee of The World Congress in Computer Science, Computer …, 66.
[43] Muhammad Faheem, Mohand-Tahar Kechadi, and Nhien An Le-Khac. 2016. The state of the art forensic techniques in mobile cloud environment: a survey, challenges and current trends. *Mobile computing and wireless networks: Concepts, methodologies, tools, and applications* (2016), 2111–2131.
[44] Ahmed Fathy, Ibrahim F Tarrad, Hesham FA Hamed, and Ali Ismail Awad. 2012. Advanced encryption standard algorithm: Issues and implementation aspects. In *Advanced Machine Learning Technologies and Applications: First International Conference, AMLTA 2012, Cairo, Egypt, December 8-10, 2012. Proceedings 1*. Springer, 516–523.
[45] Hasan Fayyad-Kazan, Sondos Kassem-Moussa, Hussin J Hejase, and Ale J Hejase. 2022. Forensic Analysis of WhatsApp SQLite Databases on the Unrooted Android Phones. *HighTech and Innovation Journal* 3, 2 (2022), 175–195.
[46] Pedro Fernández-Álvarez and Ricardo J Rodríguez. 2022. Extraction and analysis of retrievable memory artifacts from Windows Telegram Desktop application. *Forensic Science International: Digital Investigation* 40 (2022), 301342.
[47] The Electronic Frontier Foundation. 2019. Secure Messaging Scorecard. https://www.eff.org/pages/secure-messaging-scorecard. [Online; accessed 2023-01-19].
[48] Emanuele Gentili and Stefano Fratepietro. 2012. WhatsApp - Remote Change Status. https://www.exploit-db.com/exploits/18396.

[49] Henri Gilbert and Helena Handschuh. 2003. Security analysis of SHA-256 and sisters. In *International workshop on selected areas in cryptography*. Springer, 175–193.
[50] Niamh Gleeson and Ian Walden. 2016. Placing the state in the cloud: Issues of data governance and public procurement. *Computer Law & Security Review* 32, 5 (2016), 683–695.
[51] Threema GmbH. 2023. *Messenger Comparison.* https://threema.ch/en/messenger-comparison
[52] Dan Grabham. 2021. WhatsApp terms and conditions update: What you need to do and why. https://www.pocket-lint.com/apps/news/whatsapp/156398-whatsapp-terms-and-conditions-update-what-you-need-to-do-and-why/. [Online; accessed 2023-01-20].
[53] J Gregorio, B Alarcos, and A Gardel. 2018. Forensic analysis of Telegram messenger desktop on macOS. *International Journal of Research in Engineering and Science* 6, 8 (2018), 39–48.
[54] Jesús Gregorio, A Gardel, and Bernardo Alarcos. 2017. Forensic analysis of telegram messenger for windows phone. *Digital Investigation* 22 (2017), 88–106.
[55] Shay Gueron, Simon Johnson, and Jesse Walker. 2011. SHA-512/256. In *2011 Eighth International Conference on Information Technology: New Generations*. IEEE, 354–358.
[56] Phillip Hancke. 2015. Whatsapp exposed: Investigative report. 06–03 pages.
[57] Vikram S Harichandran, Frank Breitinger, Ibrahim Baggili, and Andrew Marrington. 2016. A cyber forensics needs analysis survey: Revisiting the domain's needs a decade later. *Computers & Security* 57 (2016), 1–13.
[58] Victoria Ho. 2013. Voice Messaging Comes To Whatsapp. https://techcrunch.com/2013/08/07/voice-messaging-comes-to-whatsapp/
[59] Wickr Inc. 2023. Advanced Cryptography. https://wickr.com/security/ Accessed: 2023-03-27.
[60] WhatsApp Inc. 2023. *WhatsApp Encryption Overview.* Technical Report. WhatsApp Inc. https://s3.documentcloud.org/documents/2806301/WhatsApp-Security-Whitepaper-1.pdf
[61] Wickr Inc. 2023. Wickr Me - Private Messenger. https://play.google.com/store/apps/details?id=com.mywickr.wickr2&hl=en_US&gl=US. [Online; accessed 2023-01-30].
[62] Jakob Jakobsen and Claudio Orlandi. 2016. On the CCA (in) security of MTProto. In *Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices*. 113–116.
[63] Christian Johansen, Aulon Mujaj, Hamed Arshad, and Josef Noll. 2021. The snowden phone: a comparative survey of secure instant messaging mobile applications. *Security and Communication Networks* 2021, 1 (2021), 9965573.
[64] Samantha M Judge. 2018. *Mobile Forensics: Analysis of the Messaging Application Signal.* University of Central Oklahoma.
[65] Martin Jungfer. 2021. Number of Threema users climbed to over 10 million. https://www.digitec.ch/en/page/number-of-threema-users-climbed-to-over-10-million-20061. [Online; accessed 2023-01-20].
[66] Kamil Kaczyński. 2019. Security analysis of Signal Android database protection mechanisms. *International Journal on Information Technologies and Security* 4, 11 (2019).
[67] Filip Karpisek, Ibrahim Baggili, and Frank Breitinger. 2015. WhatsApp network forensics: Decrypting and understanding the WhatsApp call signaling messages. *Digital Investigation* 15 (2015), 110–118.
[68] Giyoon Kim, Soram Kim, Myungseo Park, Younjai Park, Insoo Lee, and Jongsung Kim. 2021. Forensic analysis of instant messaging apps: Decrypting Wickr and private text messaging data. *Forensic Science International: Digital Investigation* 37 (2021), 301138.
[69] Michael D Kohn, Mariki M Eloff, and Jan HP Eloff. 2013. Integrated digital forensic process model. *Computers & Security* 38 (2013), 103–115.
[70] Jan Koum and Brian Acton. 2016. end-to-end encryption. https://blog.whatsapp.com/end-to-end-encryption.
[71] Eduard Kovacs. 2019. Signal Rushes to Patch Serious Eavesdropping Vulnerability. https://www.securityweek.com/signal-rushes-patch-serious-eavesdropping-vulnerability/.
[72] S Krishnapriya, VS Priyanka, and S Satheesh Kumar. 2021. Forensic Extraction and Analysis of Signal Application in Android Phones. In *2021 International Conference on Forensics, Analytics, Big Data, Security (FABS)*, Vol. 1. IEEE, 1–6.
[73] Manish Kumar, Akhlad Iqbal, and Pranjal Kumar. 2016. A new RGB image encryption algorithm based on DNA encoding and elliptic curve Diffie–Hellman cryptography. *Signal processing* 125 (2016), 187–202.
[74] Yesi Novaria Kunang and A Khristian. 2017. Implementasi Prosedur Forensik untuk Aplikasi Whatsapp pada Ponsel Android. *Jurnal Informatika* 11, 1 (2017), 21–35.
[75] Sebastian Kurowski. 2014. Using a whatsapp vulnerability for profiling individuals. (2014). https://dl.gi.de/bitstream/handle/20.500.12116/2633/140.pdf
[76] Ravie Lakshmanan. 2022. Critical WhatsApp Bugs Could Have Let Attackers Hack Devices Remotely. https://thehackernews.com/2022/09/critical-whatsapp-bugs-could-have-let.html. [Online; accessed 2023-01-19].
[77] Julia Len, Paul Grubbs, and Thomas Ristenpart. 2021. Partitioning oracle attacks. In *30th USENIX security symposium (USENIX Security 21)*. 195–212.
[78] Nan Li. 2010. Research on Diffie-Hellman key exchange protocol. In *2010 2nd International Conference on Computer Engineering and Technology*, Vol. 4. IEEE, V4–634.

[79] Martim Lobao. 2015. Telegram v3.2 Brings Channels For Broadcasting Your Messages To The World. https://www.androidpolice.com/2015/09/22/telegram-v3-2- brings-channels-broadcasting-messages-world/.
[80] Auqib Hamid Lone, Firdoos Ahmad Badroo, Khairaj Ram Chudhary, and Aqeel Khalique. 2015. Implementation of forensic analysis procedures for WhatsApp and Viber android applications. *International Journal of Computer Applications* 128, 12 (2015), 26–33.
[81] Joshua Lund. 2017. Encrypted profiles for Signal now in public beta. https://signal.org/blog/signal-profiles-beta/.
[82] Joshua Lund. 2018. Signal partners with Microsoft to bring end-to-end encryption to Skype. https://signal.org/blog/skype-partnership/.
[83] Aditya Mahajan, MS Dahiya, and Hitesh P Sanghvi. 2013. Forensic analysis of instant messenger applications on android devices. *arXiv preprint arXiv:1304.4915* (2013).
[84] Moxie Marlinspike. 2014. The New TextSecure: Privacy Beyond SMS. https://signal.org/blog/the-new-textsecure/.
[85] Moxie Marlinspike. 2021. Exploiting vulnerabilities in Cellebrite UFED and Physical Analyzer from an app's perspective. https://signal.org/blog/cellebrite-vulnerabilities/.
[86] Alexandre Maros, Jussara Almeida, Fabrício Benevenuto, and Marisa Vasconcelos. 2020. Analyzing the use of audio messages in Whatsapp groups. In *Proceedings of the web conference 2020*. 3005–3011.
[87] Tarun Mehrotra and BM Mehtre. 2013. Forensic analysis of Wickr application on android devices. In *2013 IEEE International Conference on Computational Intelligence and Computing Research*. IEEE, 1–6.
[88] Peter Mell and Tim Grance. 2002. *Use of the common vulnerabilities and exposures (cve) vulnerability naming scheme*. Technical Report. National Inst Of Standards And Technology Gaithersburg Md Computer Security Div.
[89] Anoshia Menahil, Waseem Iqbal, Mohsin Iftikhar, Waleed Bin Shahid, Khwaja Mansoor, and Saddaf Rubab. 2021. Forensic analysis of social networking applications on an android smartphone. *Wireless Communications and Mobile Computing* 2021, 1 (2021), 5567592.
[90] Reza Montasari, Fiona Carroll, Stuart Macdonald, Hamid Jahankhani, Amin Hosseinian-Far, and Alireza Daneshkhah. 2021. Application of artificial intelligence and machine learning in producing actionable cyber threat intelligence. In *Digital Forensic Investigation of Internet of Things (IoT) Devices*. Springer, 47–64.
[91] Mohammed Moreb. 2022. Forensic Analysis of Telegram Messenger on iOS and Android Smartphones Case Study. In *Practical Forensic Analysis of Artifacts on iOS and Android Devices: Investigating Complex Mobile Devices*. Springer, 151–193.
[92] Lindsey O'Donnell. 2020. *Android Malware Bypasses 2FA And Targets Telegram, Gmail Passwords*. https://threatpost.com/android-2fa-telegram-gmail/159384/
[93] DA Orr and A Castro. 2018. Whatsapp Messenger on the Android platform: A forensic examination of a physical device. *Forensic Science Methods and Tech* 1 (2018), 4–19.
[94] Richard E Overill. 2011. Digital Quantum Forensics: Challenges and Responses. In *Future Information Technology: 6th International Conference, FutureTech 2011, Loutraki, Greece, June 28-30, 2011, Proceedings, Part II*. Springer, 110–114.
[95] Furkan Paligu and Cihan Varol. 2020. Browser forensic investigations of whatsapp web utilizing indexeddb persistent storage. *Future Internet* 12, 11 (2020), 184.
[96] Natalie Pang and Yue Ting Woo. 2020. What about WhatsApp? A systematic review of WhatsApp and its role in civic and political engagement. *First Monday* (2020).
[97] Kenneth G Paterson, Matteo Scarlata, and Kien Tuong Truong. 2022. Three lessons from threema: Analysis of a secure messenger. *Attack Website* (2022).
[98] Trevor Perrin and Moxie Marlinspike. 2016. The double ratchet algorithm. *GitHub wiki* 112 (2016).
[99] Emmanuel S Pilli, Ramesh C Joshi, and Rajdeep Niyogi. 2010. Network forensic frameworks: Survey and research challenges. *digital investigation* 7, 1-2 (2010), 14–27.
[100] Nur Rachmat et al. 2019. Performance analysis of 256-bit AES encryption algorithm on android smartphone. In *Journal of Physics: Conference Series*, Vol. 1196. IOP Publishing, 012049.
[101] Rahadhian Dinnur Rahman and Imam Riadi. 2019. Framework Analysis of IDFIF V2 in WhatsApp Investigation Process on Android Smartphones. *Int. J. Cyber-Security Digit. Forensics* 8, 3 (2019), 213–222.
[102] Nidhi Rastogi and James Hendler. 2017. WhatsApp security and role of metadata in preserving privacy. *arXiv Prepr. arXiv1701* 6817 (2017), 269–275.
[103] Dilli Ravilla and Chandra Shekar Reddy Putta. 2015. Implementation of HMAC-SHA256 algorithm for hybrid routing protocols in MANETs. In *2015 International Conference on Electronic Design, Computer Networks & Automated Verification (EDCAV)*. IEEE, 154–159.
[104] Shripal Rawal. 2016. Advanced encryption standard (AES) and it's working. *International Research Journal of Engineering and Technology* 3, 8 (2016), 1165–1169.
[105] Ahmed Raza and Muhammad Bilal Hassan. 2022. Digital Forensic Analysis of Telegram Messenger App in Android Virtual Environment. (2022).
[106] Imam Riadi, Herman Herman, and Nur Hamida Siregar. 2022. Mobile Forensic of Vaccine Hoaxes on Signal Messenger using DFRWS Framework. *MATRIK: Jurnal Manajemen, Teknik Informatika Dan Rekayasa Komputer* 21, 3 (2022), 489–502.
[107] Marcus K Rogers and Kate Seigfried. 2004. The future of computer forensics: a needs analysis survey. *Computers & Security* 23, 1 (2004), 12–16.

[108] Paul Rösler, Christian Mainka, and Jörg Schwenk. 2018. More is less: on the end-to-end security of group chats in Signal, WhatsApp, and Threema. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 415–429.
[109] Keyun Ruan, Ibrahim Baggili, Joe Carthy, and Tahar Kechadi. 2011. Survey on cloud forensics and critical criteria for cloud forensic capability: A preliminary analysis. (2011).
[110] Daniel Ruby. 2023. 84+ Telegram Statistics In 2023 (Usage, Revenue & Facts). https://www.demandsage.com/telegram-statistics/
[111] Mr Shubham Sahu. 2014. An analysis of WhatsApp forensics in android smartphones. *International Journal of Engineering Research* 3, 5 (2014), 349–350.
[112] Fahad E Salamh, Umit Karabiyik, and Marcus K Rogers. 2020. Asynchronous forensic investigative approach to recover deleted data from instant messaging applications. In *2020 International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE, 1–6.
[113] Laura Sanchez, Cinthya Grajeda, Ibrahim Baggili, and Cory Hall. 2019. A practitioner survey exploring the value of forensic tools, AI, filtering, & safer presentation for investigating child sexual abuse material (CSAM). *Digital Investigation* 29 (2019), S124–S142.
[114] Gandeva Bayu Satrya, Philip Tobianto Daely, and Muhammad Arif Nugroho. 2016. Digital forensic analysis of Telegram Messenger on Android devices. In *2016 International Conference on Information & Communication Technology and Systems (ICTS)*. IEEE, 1–7.
[115] Paul Sawers. 2015. Three-quarters of WhatsApp users are on Android, 22% on iOS (study). https://venturebeat.com/mobile/three-quarters-of-whatsapp-users-are-on-android-study-finds/
[116] Joost Schellevis. 2014. WhatsApp-status van anderen is nog steeds te wijzigen. https://tweakers.net/nieuws/79321/whatsapp-status-van-anderen-is-nog-steeds-te-wijzigen.html. [Online; accessed 2023-01-19].
[117] Svenja Schröder, Markus Huber, David Wind, and Christoph Rottermanner. 2016. When SIGNAL hits the fan: On the usability and security of state-of-the-art secure mobile messaging. In *European Workshop on Usable Security. IEEE*. 1–7.
[118] Seth. 2023. *Does Wickr log or track my communications or activity?* https://support.wickr.com/hc/en-us/articles/115007884828-Does-Wickr-log-or-track-my-communications-or-activity- Accessed: 2023-03-27.
[119] Ax Sharma. 2021. Researcher refuses Telegram's bounty award, discloses auto-delete bug. https://arstechnica.com/information-technology/2021/10/researcher-refuses-telegrams-bounty-award-discloses-auto-delete-bug/. [Online; accessed 2023-01-19].
[120] Adam Shortall and MA Hannan Bin Azhar. 2015. Forensic acquisitions of WhatsApp data on popular mobile platforms. In *2015 Sixth International Conference on Emerging Security Technologies (EST)*. IEEE, 13–17.
[121] Catherine Shu. 2013. Meet Telegram, A Secure Messaging App From The Founders Of VK, Russia's Largest Social Network. https://techcrunch.com/2013/10/27/meet-telegram-a-secure-messaging-app-from-the-founders-of-vk-russias-largest-social-network/. [Online; accessed 2023-01-19].
[122] C Shubha, SA Sushma, and KH Asha. 2019. Traffic analysis of whatsapp calls. In *2019 1st International Conference on Advances in Information Technology (ICAIT)*. IEEE, 256–260.
[123] Jihun Son, Yeong Woong Kim, Dong Bin Oh, and Kyounggon Kim. 2022. Forensic analysis of instant messengers: Decrypt Signal, Wickr, and Threema. *Forensic Science International: Digital Investigation* 40 (2022), 301347.
[124] Nishchal Soni. 2024. Forensic Analysis of WhatsApp: A Review of Techniques, Challenges, and Future Directions. (2024).
[125] Wickr Staff. 2022. Our focus on end-to-end encrypted enterprise communications. https://wickr.com/our-focus-on-end-to-end-encrypted-enterprise-communications/
[126] Bernd Carsten Stahl, Job Timmermans, and Brent Daniel Mittelstadt. 2016. The ethics of computing: A survey of the computing-oriented literature. *Acm Computing Surveys (CSUR)* 48, 4 (2016), 1–38.
[127] EK Subramanian and Latha Tamilselvan. 2020. Elliptic curve Diffie–Hellman cryptosystem in big data cloud security. *Cluster Computing* 23, 4 (2020), 3057–3067.
[128] Tomáš Sušánka and Josef Kokeš. 2016. Security Analysis of the Telegram IM. In *Proceedings of the 1st Reversing and Offensive-oriented Trends Symposium* (Vienna Austria, 2017-11-16). ACM, 1–8. doi:10.1145/3150376.3150382
[129] The Telegram Team. 2001. Crowdsourcing a More Secure Future. https://telegram.org/blog/crowdsourcing-a-more-secure-future.
[130] The Telegram Team. 2023. The Evolution of Telegram. https://telegram.org/evolution.
[131] Neha S Thakur. 2013. Forensic analysis of WhatsApp on Android smartphones. (2013).
[132] Helmy Trisnasenjaya and Imam Riadi. 2019. Forensic Analysis of Android-based WhatsApp Messenger Against Fraud Crime Using The National Institute of Standard and Technology Framework. *International Journal of Cyber-Security and Digital Forensics* 8, 1 (2019), 89–98.
[133] William Turton. 2016. Why You Should Stop Using Telegram Right Now. https://gizmodo.com/why-you-should-stop-using-telegram-right-now-1782557415. [Online; accessed 2023-01-19].
[134] Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith. 2015. SoK: secure messaging. In *2015 IEEE Symposium on Security and Privacy*. IEEE, 232–249.
[135] Andrea Visconti, Simone Bossi, Hany Ragab, and Alexandro Calò. 2015. On the weaknesses of PBKDF2. In *Cryptology and Network Security: 14th International Conference, CANS 2015, Marrakesh, Morocco, December 10-12, 2015, Proceedings 14*. Springer, 119–126.
[136] Rizaldi Wahaz, Rakha Nadhifa Harmana, Amiruddin Amiruddin, and Ardya Suryadinata. 2021. Is whatsapp plus malicious? a review using static analysis. In *2021 6th International Workshop on Big Data and Information Security (IWBIS)*. IEEE, 91–96.

[137] Daniel Walnycky, Ibrahim Baggili, Andrew Marrington, Jason Moore, and Frank Breitinger. 2015. Network and device forensic analysis of android social-messaging applications. *Digital Investigation* 14 (2015), S77–S84.
[138] Stiftung Warentest. 2022. Messenger im Vergleich. https://www.test.de/Messenger-Apps-im-Vergleich-4884453-0/. [Online; accessed 2023-01-19].
[139] Takuya Watanabe, Mitsuaki Akiyama, Fumihiro Kanei, Eitaro Shioji, Yuta Takata, Bo Sun, Yuta Ishi, Toshiki Shibahara, Takeshi Yagi, and Tatsuya Mori. 2017. Understanding the Origins of Mobile App Vulnerabilities: A Large-Scale Measurement Study of Free and Paid Apps. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)* (2017-05). 14–24. doi:10.1109/MSR.2017.23
[140] Zack Whittaker. 2017. DocumentCloud. https://www.documentcloud.org/documents/3723701-Ron-Wyden-letter-on-Signal-encrypted-messaging.html.
[141] I Gusti Ngurah Guna Wicaksanaa and I Ketut Gede Suhartanaa. 2020. Forensic Analysis of Telegram Desktop-based Applications using the National Institute of Justice (NIJ) Method. *Jurnal Elektronik Ilmu Komputer Udayana p-ISSN* 2301 (2020), 5373.
[142] Jan Wichelmann, Sebastian Berndt, Claudius Pott, and Thomas Eisenbarth. 2021. Help, My Signal has Bad Device!. In *Detection of Intrusions and Malware, and Vulnerability Assessment* (Cham, 2021) *(Lecture Notes in Computer Science)*, Leyla Bilge, Lorenzo Cavallaro, Giancarlo Pellegrino, and Nuno Neves (Eds.). Springer International Publishing, 88–105. doi:10.1007/978-3-030-80825-9_5
[143] WickrInc. 2021. GitHub - WickrInc/wickr-crypto-c: An implementation of the Wickr Secure Messaging Protocol in C. https://github.com/WickrInc/wickr-crypto-c
[144] Dennis Wijnberg and Nhien-An Le-Khac. 2021. Identifying interception possibilities for WhatsApp communication. *Forensic Science International: Digital Investigation* 38 (2021), 301132.
[145] Jeremy Wohlwend. 2016. *Elliptic curve cryptography: Pre and post quantum.* Technical Report. Technical report.
[146] Samarjeet Yadav, Satya Prakash, Neelam Dayal, and Vrijendra Singh. 2019. Forensics analysis of whatsapp in android mobile phone. In *Proceedings of the international conference on advances in electronics, electrical & computational intelligence (ICAEEC).*
[147] Shuo Yan, Kim-Kwang Raymond Choo, and Nhien-An Le-Khac. 2022. Signal Instant Messenger Forensics. In *A Practical Hands-on Approach to Database Forensics*. Springer, 27–92.
[148] Fietyata Yudha, Ahmad Luthfi, and Yudi Prayudi. 2017. A proposed model for investigating on web WhatsApp application. *Advanced Science Letters* 23, 5 (2017), 4050–4054.
[149] Vindy Arista Yuliani and Imam Riadi. 2019. Forensic Analysis WhatsApp Mobile Application On Android-Based Smartphones Using National Institute of Standard and Technology (NIST) Framework. *vol* 8 (2019), 223–231.

Received 25 May 2023; revised 6 March 2025; accepted 22 March 2025