Received: 3 June 2023 | Revised: 22 March 2024 | Accepted: 22 April 2024

DOI: 10.1002/spe.3341

RESEARCH ARTICLE

WILEY

Threat models over space and time: A case study of end-to-end-encrypted messaging applications

Partha Das Chowdhury[1] | **Maria Sameen**[1] | **Jenny Blessing**[2] | **Nicholas Boucher**[2] | **Joseph Gardiner**[1] | **Tom Burrows**[2] | **Ross Anderson**[2,3] | **Awais Rashid**[1]

[1]School of Computer Science, University of Bristol, Bristol, UK

[2]Computr Laboratory, University of Cambridge, Cambridge, UK

[3]School of Informatics, University of Edinburgh, Edinburgh, UK

Correspondence
Partha Das Chowdhury, School of Computer Science, University of Bristol, Merchant Ventures Building, 75 Woodland Road, Bristol BS8 1UB, UK.
Email: partha.daschowdhury@bristol.ac.uk

Funding information
REPHRAIN: Research Centre on Privacy, Harm Reduction and Adversarial Influence Online, Grant/Award Number: EPSRC: EP/V011189/1

Abstract

Threat modeling is one of the foundations of secure systems engineering and must take heed of the context within which systems operate. In this work, we explore the extent to which real-world systems engineering reflects a changing threat context. We examine the desktop clients of six widely used end-to-end-encrypted mobile messaging applications to understand the extent to which they adjusted their threat model over space (when enabling clients on new platforms, such as desktop clients) and time (as new threats emerged). We experimented with short-lived adversarial access against these desktop clients and analyzed the results using two popular threat elicitation frameworks, STRIDE and LINDDUN. The results demonstrate that system designers need to track threats in the evolving context within which systems operate and, more importantly, mitigate them by rescoping trust boundaries so that they remain consistent with administrative boundaries. A nuanced understanding of the relationship between trust and administration is vital for robust security, including the provision of safe defaults.

KEYWORDS

end to end encrypted communications, security and privacy threat modeling, security engineering

1 | INTRODUCTION

Threat modeling has become an integral part of secure software development. Using a framework such as STRIDE (spoofing, tampering, repudiation, information disclosure, denial of service, elevation of privilege) is a key step within the Microsoft Security Development Lifecycle (SDL)[1] to analyze application information flows against key classes of threats. Threat modeling is also a recommended best practice by OWASP[2] and within Agile[3] and DevOps processes.[4] Researchers have developed similar frameworks to systematically analyze threats to user privacy when developing software applications.[5]

However, threat modeling cannot be a one-off activity. Any entity doing so as a one-off activity is not following documented SDL best practices as threats evolve and new attacks come to light, developers must continuously reassess

Abbreviations: E2EE, end-to-end-encrypted; ID, identity; LINDDUN, linking, identifying, non-repudiation, detecting, data disclosure, unawareness, and non-compliance; ops, operations; QA, quality assurance; SDL, software development lifecycle; STRIDE, spoofing, tampering, repudiation, information disclosure, denial of service and elevation of privilege; TM, threat model; US, United States.

applications against them. Furthermore, adding new features to an application creates new information flows, which in turn can cause trust boundaries to shift, as can also happen as systems acquire additional hardware or software components or third-party services. The recommended best practice is to do threat modeling "little and often."[6] We focus on the effects of a dynamic threat landscape as new features are added to existing applications.

We use the term space to denote the addition of new features to existing applications. Threat landscape on the other hand is ever evolving. Prior research asserted that it is difficult for application developers to account for all possible vulnerabilities.[7] Moreover, studies with application developers have highlighted the difficulty of threat modeling in agile environments, and suggested actionable interventions to aid secure development.[8,9] A longitudinal study of 675,000 application developers on their use of software security activities, conclude that only 1% of their participants *build/use abuse cases in QA process*.[10] On the other hand the same study concludes that *bugs in OPS fed back to dev/change behavior*. Situating explicit considerations of space and time in the development process would provoke application developers to test abuse cases as they emerge with time on the space of expanded feature sets. We establish the concepts of space and time through a case study—whether the threat models that underpin security and privacy aspects of applications evolve through space and time.

Our case study consists of end-to-end-encrypted (E2EE) mobile messaging applications (such as Signal[*] and WhatsApp), which have found widespread adoption amongst users aiming to protect the privacy of their communications and mitigate large-scale surveillance.[11,12] E2EE mobile apps were conceived to protect message contents from potential eavesdroppers on the communication channel (which includes the messaging provider itself). Most of these platforms have since also launched desktop clients to make it easier for users to handle larger, more complex messages, and communicate across multiple devices. Introduction of desktop clients represent the space of expanded feature set of the existing mobile messaging applications.

In the meantime, security and privacy threats against which users need to be protected have also evolved. For instance, the mobile app messaging threat model was largely predicated on an eavesdropper on the communication channel. However, research into intimate partner violence has highlighted that abusers often use monitoring technology or shared devices to surveil and exert control over victims.[13,14] In this scenario, the threat actor is not remote but has direct physical access to the victim's devices. For an abuse victim, the persistence of access, which may last long after they have left their abuser, is of serious concern.

There are other contexts in which even short-lived access to a desktop client can pose potential threats:

- **Official searches:** Border and customs officials sometimes search travellers' devices and make copies of data stored on them. Potential exploitation of the access to E2EE messaging enabled by desktop clients would pose a serious threat to activists and journalists—a major concern in the context of the Russian invasion of Ukraine and the use of E2EE messaging applications to coordinate activities by defenders and for protecting free speech. The leakage of medical communications is a further concern following the overturning of Roe v. Wade in the US.
- **Shared devices:** Many households share devices, especially desktop computers and laptops. There is a high likelihood that individuals living in the same household could gain at least temporary access to each other's machine or device backups. The security and privacy provided by E2EE messaging should avoid leakage of sensitive information in domestic contexts where there may be strife or abuse.
- **Corporate Machines:** Machines provided by an employer are often managed remotely, with company sysadmins having full access. Any desktop clients of E2EE messaging applications must ensure that the communications are protected from adversarial sysadmins.

Figure 1 represents a conceptualization of the importance of evolving threat models over space and time. In order to explore if this happens in practice, we systematically analyze six major E2EE messaging applications: Signal, WhatsApp, Element, Viber, Wickr Me, and Telegram. We start from the original threat model (TM_1) of these applications, that is, a mobile app client S_1 with a remote attacker at time t_1. We then develop a second (TM_2) at the intersection of the expanded feature space that is, the desktop client in its operating context (shared and/or managed device) S_2 at time t_2. Using an experimental test setup, we then simulate adversarial short-lived access (TM_2) to the desktop clients of each of the six applications. The resultant compromise (if any) is mapped with respect to security (using the STRIDE threat modeling

[*] We refer to the Signal messenger application here, rather than the Signal protocol, and will continue to distinguish between the two throughout the article.

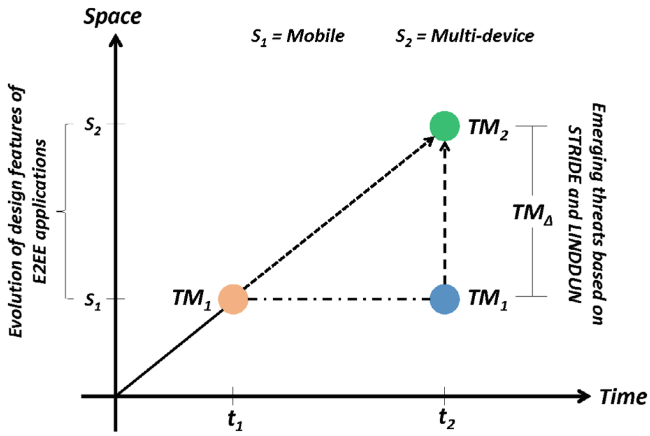

FIGURE 1 Evolution of threat models.

approach[1]) and privacy (using the LINDDUN threat modeling approach[5]) to elicit security and privacy threats respectively. The resultant security and privacy threats comprise the net evolution (TM_Δ) for each application. Our analysis shows the applications evolve their threat model to varying degrees to mitigate the threats resulting from such adversarial short-lived access.

We argue based on our investigation that threat models (and hence protection mechanisms informed by them) need to evolve in space and time as threats change. For some desktop clients, TM_Δ reveals their vulnerability to *spoofing, repudiation, information disclosure and elevation of privilege* in the face of short-lived adversarial access. These vulnerabilities, in turn, give rise to two kinds of privacy leakage: *linkability* of information and *identifiability* of the communicating parties. Our analysis also highlights that recognition of the change in the threat context is useful but not enough in itself unless backed by appropriate countermeasures.

2 | RELATED WORK

There is a sizable body of work investigating and demonstrating vulnerabilities in E2EE messaging protocols and their implementations. Prior work has focused particularly on Telegram due to its use of a custom protocol, and researchers have demonstrated numerous attacks over the years on both the MTProto protocol and its implementation.[15,16] Most recently, the Matrix protocol (which Element uses) suffered several severe vulnerabilities in which a malicious actor was able to break the confidentiality of communications using a compromised Matrix server.[17]

Research has also demonstrated the technical feasibility of device cloning as a means of accessing user accounts on E2EE messaging platforms. Cremers et al.[18] investigated practical post-compromise security measures in the mobile clients and some desktop clients of the main E2EE messaging platforms, finding that almost all clients studied are vulnerable to some degree of cloning attack. The authors identify challenges in consistently maintaining state across devices as the primary practical reason why these platforms do not attempt to prevent cloning: a post-compromise security mechanism can run the risk of being overly strict (cf. Figure 7B and lock legitimate users out of their devices if synchronization fails, leading to usability concerns. However, our experiments show that the desktop clients of Viber, WickrMe and Element do better in detecting cloning attempts than their mobile counterparts.[18] Cisco's threat intelligence unit further experimented with cloning a handful of E2EE desktop clients (specifically, Signal, WhatsApp, and Telegram) in the wake of the 2018 "Telegrab" malware strain that took advantage of Telegram's vulnerability to desktop cloning.[19] Similarly to Cremers et al., they found that all three desktop clients studied allowed a cloned client to masquerade as legitimate, sending and receiving messages with little to no indication to the original user.

In this article, we contribute to this area by analyzing the problem from a threat modeling perspective. We demonstrate that the problem arises from a lack of consideration of threats across space and time. The evolution of applications is a reality as is the addition of new features. However, rescoping the trust boundary as both the application and the threats evolve is critical. Our analysis of the delineation of trust and administrative boundaries provides a basis for maintaining security as systems and their environments evolve and at the tactical level for better administration of shared components, including the design of safe defaults.

3 | BACKGROUND: E2EE MESSAGING APPLICATIONS AND THREAT FRAMEWORKS

In this section, we tease out the root assumptions underpinning the security of the E2EE applications we investigate. We summarize the key security properties of these applications and their desktop clients in Table 1. This is followed by a description of the security and privacy threat modeling frameworks we use for this investigation—STRIDE[1] and LINDDUN.[5]

3.1 | E2EE messaging applications

Every installation is tied to a particular user identity. This identity is then used as the root of trust to communicate securely with other participants through the service and to configure additional devices for the same account. The steps taken to generate a long-term identity key and corresponding short-lived asymmetric key pairs are as follows:

1. Every communication node generates its own long-term public-private key pair—the identity key *IK*. Any principal that can prove possession of the secret component of *IK* is considered the legitimate owner of the account connected to this identity key.
2. Ephemeral asymmetric key pairs known as pre-keys help encrypt messages between communicating entities. The goal is to assure forward and backward secrecy even if one or more pre-keys are compromised.
3. The public components of the pre-keys are signed using the long-term identity key and communicated via the server to other communicating entities. The assumption is that only the account owner can create such signatures.
4. The private components of *IK* and the pre-keys are not known to anyone else, including the application provider server. Thus the communications are encrypted end-to-end.

3.1.1 | Mobile applications

Signal, WhatsApp, Element and Viber broadly adhere to the Signal protocol's Double Ratchet algorithm,[20] as shown in Table 1. In essence, every communication sequence is encrypted with the pre-key listed against the intended recipient. The process of deriving secrets from previously held secrets is termed as ratcheting and is initialized with a shared secret.

TABLE 1 Properties of popular messaging applications.

Applications	Protocol	Primary device (phone) parameters	Desktop client
Signal	Signal	Curve25519 key pair—long term identity key Curve25519 key pair—pre-keys	Desktop ID authenticated by primary device Can be used independently
WhatsApp	Signal	Curve25519 key pair—long term identity key Curve25519 key pair—pre-keys	Desktop ID authenticated by primary device Can be used independently
Element	Olm-Double Ratchet implementation	Curve25519 key pair—long term identity key Curve25519 key pair—pre-keys	Desktop ID authenticated by primary device Can be used independently
Wickr Me	Wickr secure messaging protocol	Curve P521 key pairs SHA-256 device identifier	Desktop ID authenticated by primary device Can be used independently
Viber	Double Ratchet implementation	Curve25519 key pair—long term identity key	Desktop client authenticated by primary device Can be used independently
Telegram	MTProto 2.0—Diffie Hellman implementation	Cloud chat—2048 bit permanent key Secret chat—DH keys between communicating entities	Desktop ID authenticated by primary device Can be used independently

Sequence numbers help to decrypt out-of-order messages. The root of trust for each participant is the identity key they configured during account setup.

Wickr Me uses the Wickr secure messaging protocol,[21] which pins key material to identifiers generated by devices, typically in the form of SHA-256 values. The application/device is the primary actor in this protocol and the basis of all trust relationships. The important primitives are the identity keys *IK*, ephemeral asymmetric key pairs (similar to pre-keys) and a platform-specific message encryption key.

Telegram is a messenger based on MTProto 2.0. Client and server share a device-specific 2048-bit permanent authorization key created by a Diffie-Hellman (DH) key exchange.[22] Communication can happen through a cloud server, or users can switch to secret chat, giving E2EE communication where short-lived keys are linked to the device-specific permanent authorization key.

3.1.2 | Desktop clients

Many E2EE apps offer a desktop client that can piggyback on the E2EE functionality of the phone or other primary device. These clients share some common characteristics:

- Every user needs to install the mobile application on a primary device which they control and have an account through it on the messenger service.
- After installation, the desktop client for a messaging application will typically generate its own identity key pair, distinct from that of the primary device.
- The primary device and the desktop client authenticate each other. The primary device then tells the application server that the desktop client is trusted and can communicate as if it were the primary device.
- The primary device retains a list of the linked companion devices.

All the E2EE applications we investigated assume that account holders will be able to protect the private component of their identity key. The documentation does not typically give any advice as to how; many providers of messaging services consider endpoint security to be out-of-scope. This sits uncomfortably with the fact that mechanisms such as ratchets are designed to recover from transient device compromise by supporting forward and backward security properties. We elaborate on this further in our analysis.

3.2 | Threat modeling

Threat modeling is a key enabler for security and privacy by design.[23] As a case in point, Microsoft's secure development lifecycle relies on it.[1] Its approach to systematic elicitation of security threats to the data flows between various system components is known as STRIDE. A brief description of various threat modeling methods can be found in Reference 24. Some threats to privacy are distinct from those in STRIDE, and so a second methodology, LINDDUN, captures threats to privacy, such as the ability of an adversary to link actions to principals and thus defeat anonymity.

The configuration of desktop clients involve flow of secret information from the primary devices to companion devices and a shift in system boundary. STRIDE and LINDDUN both model systems in terms of boundaries and flows; which we believe is appropriate to capture the security and privacy threats in our example scenario. Moreover they do so using industry-standard tools.

3.2.1 | STRIDE

In STRIDE, the target system is modeled using data flow diagrams (DFDs) that capture the key components (processes and data stores), and the data flows between them. Trust boundaries mark which parts of the system are assumed to be free from adversarial interference. Data flows across trust boundaries and individual components are then evaluated for their susceptibility to six key threats:

1. Spoofing manifests as a negation of the security property *authentication*. Such threats enable attackers to masquerade as a legitimate user.
2. Tampering results due to violation of *integrity*. This threat leads to unauthorized modification of data.
3. Repudiation threats result due to failure of the security property of *non-repudiation*. Such threats make it impossible to gather irrefutable evidence against mal-actors.
4. Information disclosure threats violate the security property *confidentiality*. They allow unauthorized exposure of confidential information.
5. Denial of service results due to malicious compromise of the security property of *availability*. Consequently, legitimate users are unable to access or use a system.
6. Elevation of privilege threats violate the security property *authorization*. They allow attackers to gain sufficient access to information and resources to compromise them.

3.2.2 | LINDDUN

LINDDUN follows a similar approach to STRIDE but its focus is on threats to privacy. The system is once more modeled using DFDs, but in this case, the individual DFD elements and data flows are evaluated for seven privacy threats. They are as:

1. Linkability threats enable an adversary to violate the privacy property *unlinkability*, thereby correlating otherwise distinct items like subjects, messages, actions and so forth.
2. Identifiability results as a negation of the privacy property *anonymity/pseudonymity*. While the previous threats allow an adversary to link subjects to actions, this threat enable complete identification of the subject.
3. Non repudiation emerges as a privacy threat violating the privacy property *repudiation*. This allow unauthorized users to repudiate claims by being able to link actions to subjects irrefutably.
4. Detectability enables a malicious entity to compromise *undetectability/unobservability* of their victims. This means an observer can sufficiently conclude about the existence of an item of (their) interest.
5. Information disclosure violates *confidentiality* property. This threat would allow an adversary to gain access to personal sensitive information of their victims.
6. Content unawareness results due to negation of the *awareness* property. Such threats leave users in the dark about what information about them is disclosed to the system and such information can potentially be misused.
7. Policy and noncompliance results due to violation of *consent and compliance* policies. Such threats imply that systems can misuse personal sensitive information and even expose sensitive information to unauthorized entities.

4 | METHODOLOGY

4.1 | Choice of E2EE applications

We selected applications that are widely used and diverse in how they establish trust between primary and companion devices. They can be divided into two broad categories: those that rely on variants of the Signal protocol, and those that do not. For Signal-based apps, we study two implementations, Signal's and WhatsApp's; Signal's is open-source while WhatsApp's is not. As we will note later, there are noticeable differences. While not relying directly on the Signal protocol, Viber and Element both make use of its Double Ratchet algorithm. Viber differs from other implementations in the way the Root ID is shared between the primary and companion devices. We examine whether this affects the ability to protect against threats from short-lived adversarial access. Element is noteworthy for being decentralized; it does not rely on a central communications server. We investigate if this has any bearing on trust establishment between companion and primary devices.

Two further messaging services, Wickr Me and Telegram, rely entirely on their own messaging protocols. Wickr documentation indicates that device-specific information is used in device enrolment, so we evaluate whether this is sufficient to prevent silent desktop cloning. Telegram uses a custom protocol that distinguishes between "cloud" chats and "secret" chats. The documentation does not discuss any measures for forward secrecy post-compromise.

4.2 | Creation of DFDs

We created DFDs (using Microsoft's Threat Modeling Tool) for each app before and after the addition of the desktop client, first by studying their documentation[11,21,25–28] and second, through experiments (see Section 4.4). For example, WhatsApp documentation explicitly states[25(p. 25)]

> WhatsApp defines end-to-end encryption as communications that remain encrypted from a device controlled by the sender to one controlled by the recipient, where no third parties, not even WhatsApp or the parent company Facebook, can access the content in between.

The red dotted line indicates the trust boundary that demarcates the security-critical artifacts. Figure 2 depicts the DFD at time t_1 and space S_1 (cf. Figure 1) for the six mobile messaging applications. The private part of the *IK* and the pre-keys are inside the mobile device in which they were generated. Operation without a linked desktop client does not entail sharing them with any other device. There the designers consider that the secrets, being in the device, are under the control of their owners. The documentation advises that any device compromise should be reported and credentials revoked. So we place the security artifacts within the red trust boundary, keeping the eavesdropper outside.

Figures 3 and 4 show the DFD at time t_2 and space S_2 (cf. Figure 1) where a desktop client has been added. The difference in the way companion devices are set up leads to two different DFDs. While all the desktop clients are first launched using the primary mobile device to which they are linked, there are then two separate cases. For Signal, WhatsApp and Telegram subsequent desktop clients can be launched without using the primary mobile device, but this is not the case for Viber, Element and WickrMe.

The red dotted line in Figure 3 represents the trust boundary and thus the placement of security artifacts for Signal, WhatsApp and Telegram. The private part of *IK* and the pre-keys are within the realm of everyone with access to the desktop.

Figure 4 represents the placement of the security artifacts for Viber, Element and WickrMe. The trust boundary includes the private part of *IK* and the pre-keys. The reason is that every instance of a desktop client needs to be explicitly launched by the principal mobile device. As the keys cannot be copied, desktop clients cannot be launched independently.

The shift in the trust boundary is determined by our findings. Instances where security critical artifacts from the companion device are not accessible or cannot be used by an adversary without the explicit consent of the legitimate

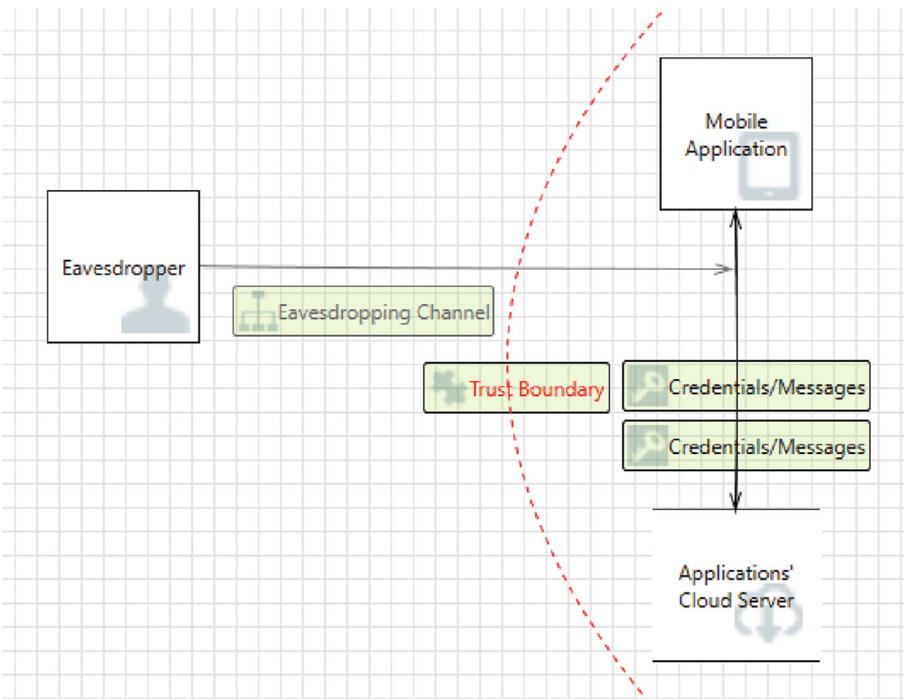

FIGURE 2 DFD for the Signal, WhatsApp, Element, Wickr Me, Viber, and Telegram mobile applications.

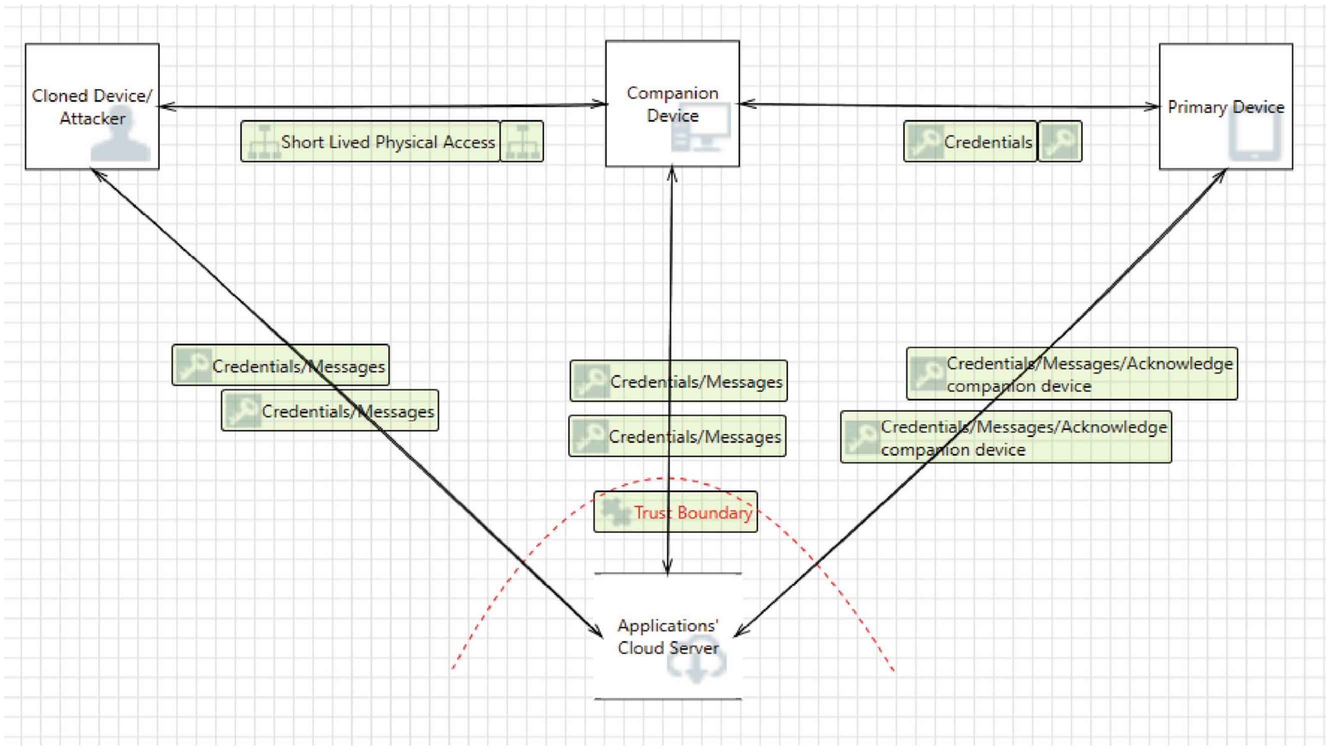

FIGURE 3 DFD for the Signal, WhatsApp, and Telegram desktop applications.

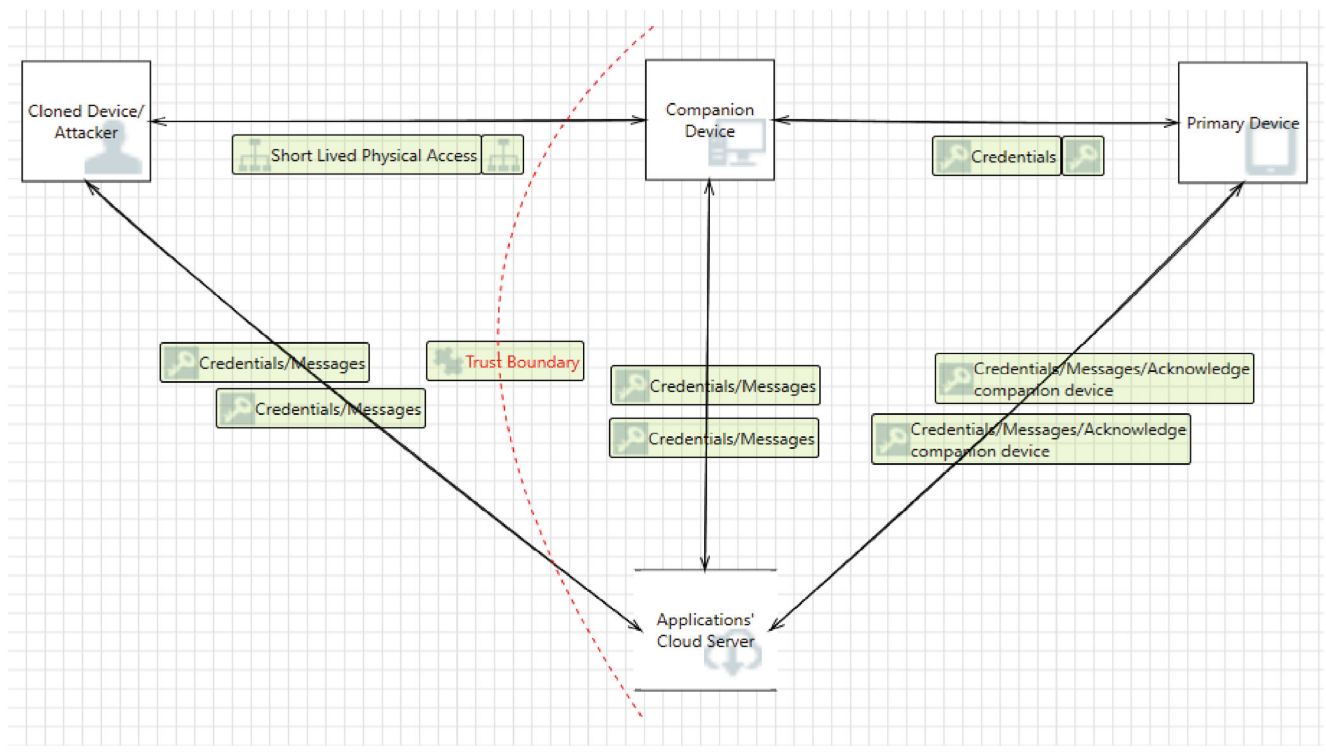

FIGURE 4 DFD for the Element, Wickr Me and Viber desktop applications.

owner are expressed by placing the companion device within the trust boundary. Where, security critical artifacts from the companion device are accessible or can be used from the companion device, the companion device is placed outside the trust boundary.

4.3 | Experiment goals

Our goal was to find TM_Δ for each of the desktop clients. TM_Δ consists each element of the threats we consider for this article—if a desktop client is vulnerable to spoofing then $\{TM_\Delta = \text{spoofing}\}$. We report the findings in Section 5. The experimental set up and process to test each of the threats are discussed in Sections 4.4 and 4.5 respectively.

4.4 | Experimental setup

The experiments were conducted between test accounts registered to phone numbers provided by pay-as-you-go SIM cards purchased specifically for these experiments. Account registration was performed using a Samsung Galaxy A21 smartphone and iPhone SE which were used to receive SMS messages required for registration. The desktop clients were installed through the following steps:

1. **Hardware**. The desktop clients were installed on MacBook Pro laptops with 2 GHz Quad-Core Intel Core i5 and 16 GB 3733 MHz LPDDR4X memory.
2. **Firing up the legitimate desktop client**. We started a legitimate desktop version of the application through the required setup mechanism, for example, typically by scanning a `QR code`.
3. **Firing up the attacker's desktop client**. We performed a standard installation of the desktop client on an attacker's machine, configured with a second account, then copied the state from the victim's machine and placed it in the attacker's. Our goal was to evaluate if these systems protect against simple cloning attacks, so we copied state information from `~/Library/Application Support/` of the victim's machine to the same directory of the attacker's machine. The victim's machine and the attacker's machine were of the same specification.

4.5 | Testing for threats

We use two threat frameworks to distinctly draw the security and privacy threats. Short lived adversarial access results due to *authentication* and *authorization* failures; they are security properties with privacy consequences. While STRIDE captures the security threats like *spoofing*, it is unable to capture the privacy consequences on the victim (whose desktop client is compromised) and the communicating entities connected to the victim. LINDDUN captures the privacy consequences like *linkability* and *identifiability*.

4.5.1 | STRIDE

- **Spoofing**. We performed a standard installation of the desktop client in an attacker's machine, then copied the state from the victim's machine to the attacker's.
- **Tampering**. Our focus was on the endpoints rather than the network, so we did not experiment with altering message content while in transit.
- **Repudiation**. While we set up the attacker's machine, we engaged in communication between the legitimate participant, the attacker and an innocent third party. This was repeated between the communicating entities to understand if third parties observe any difference when communicating with the victim and the attacker.
- **Information disclosure**. We used the cloned desktop across space and time to understand the implications of forward and backward secrecy.
- **Denial of service**. We tested whether the cloned machine throws the victim out of the network or allows the victim to continue sending and receiving messages even when the clone is in operation.

- **Elevation of privilege**. This was tested by capturing the credentials using a `tls interceptor` from the rooted device. Then the victim's desktop client was de-linked from the primary device. Subsequently, the cloned desktop client was also de-linked. Then we used the captured credentials to restart the desktop client in the attacker's device.

4.5.2 | LINDDUN

We focused on identifiability and linkability as information disclosure and non-repudiation versus repudiation were already evaluated in our STRIDE analysis. Since our focus was on endpoints, we did not perform an analysis for detectability. Non-compliance and unawareness are out of our scope for us—we investigate the vulnerabilities of the registration processes and not their gaps with respect to regulation and information sharing with their users. Privacy vulnerabilities, however can provoke appropriate regulations and compliance needs.

- **Linkability**: The various artifacts from a victim were checked if they link potential entities connected to the victim.
- **Identifiability**: We checked if those artifacts revealed identifying information about the victims and indirect entities connected to the victim.

We then analyzed the type(s) of data accessible through potential threats. We then modeled these data items as trees to depict the identifiability and linkability of an entity.

5 | FINDINGS

A longstanding security goal in establishing a communication channel is to ensure that the only participants in the channel are legitimate ones. So desktop clients of E2EE messaging apps should be capable of identifying distinct participants in a channel, and thereby withstand cloning attacks following short-lived adversarial access attacks. We, therefore, subject the desktop client to short-lived adversarial access (TM_2) to elicit the threats based on the tests particular to STRIDE and LINDDUN.

Table 2 shows the threats which were not scoped while expanding from S_1 to S_2 between time t_1 and t_2. Each cell indicates if a particular threat was tested on the desktop clients and the test result. Section 4.5 enumerates the threats we consider for this work and (-) indicates threat not being tested. For the ones we test (×) indicates attack not possible; and (✓) indicates attack is possible. For example, our experiments show that an adversary can clone a Signal desktop client and compromise forward secrecy. We indicate the consequences of this risk in Table 2 against the specific security and privacy threats—(✓) against *STRIDE—spoofing, repudiation, information disclosure and elevation of privilege* and *LINDDUN—linkability, identifiability, non-repudiation and information disclosure*. The same applies for all the desktop clients and threat intersection cell values in the table.

TABLE 2 (TM_{Δ}) based on STRIDE and LINDDUN threat models.

	Emerging threats (TM_{Δ})											
Applications	**S**	**T**	**R**	**I**	**D**	**E**	**L**	**I**	**N**	**D**	**D**	**U**	**N**
Signal	✓	-	✓	✓	×	✓	✓	✓	✓	-	✓	-	-
Whatsapp	✓	-	✓	✓	×	×	✓	✓	✓	-	✓	-	-
Element	×	-	×	✓	×	×	✓	×	×	-	✓	-	-
Wickr Me	×	-	×	×	×	×	×	×	×	-	×	-	-
Viber	×	-	×	×	×	×	×	×	×	-	×	-	-
Telegram	✓	-	✓	✓	×	×	✓	✓	✓	-	✓	-	-

Note: In this table, (-) indicates not being tested; (×) indicates attack not possible; and (✓) indicates attack is possible.

5.1 | Signal messenger

Signal messenger assumes that only an eavesdropper can be, or attempt to be, the adversary and implements its authentication and key-sharing mechanisms accordingly. Against other adversaries, the expectation is that the user would replace the device or account. This may have been appropriate before companion devices were supported, as private keys never left the primary device. However, our experiments show that such assumptions crumble when potential adversaries reside within the trust boundary and adversarial short-lived access can go undetected.

Masquerading as the victim: An attacker can simply replace the configuration files of a standard Signal desktop installation with the version stolen from a victim's machine. The specifics of the attacker machine do not influence the success of the attack. Signal desktop uses an encrypted SQLite database to store Signal authentication credentials: login and device password (the `uuid_id` and `password` fields), received messages, and pre-keys. The relevant files are named `config.json` and `databases/Databases.db`. The database decryption key is stored in plaintext in the parent directory in `config.json`. For a knowledgeable adversary, the database is effectively stored in plaintext.† The collapse of the assumption of an *only eavesdropper* threat model leads to the violation of other security properties with privacy consequences.

When the Signal application is re-installed on a device, whether due to device compromise or for other reasons such as a new phone, the user identity key changes along with the pre-keys. This is notified to all their contacts, alerting them to the change and enabling them to verify new keys out-of-band if they wish. However, a comparison between the legitimate desktop version and a cloned desktop shows the same keys against the sequence number of the pre-keys. This brings home that the DH ratchet is not effective at rendering the cloned version obsolete after the existing key material is exhausted.

The mobile application assumes that the security protocol mechanisms will prevent an eavesdropper from capturing enough state information to masquerade as the victim—there are message sequence numbers to prevent an eavesdropper from simply replaying past messages. In such a case, improper communication will be detected automatically. There is no such protection against an adversary with short-lived access to the desktop client. There are still overt symptoms that could be noticed by an alert victim. For example, the attacker's Signal desktop may work with delays or messages will be dropped when the victim's mirrored desktop installation is actively online. In none of our experiments, the active desktop thrashed visibly between the original and the clone. Our observation was that such behavior was dependent on whether the session was established by the victim, the attacker or the communicating second party. On subsequent examination, we noted that the Signal protocol documentation[20] advises setting a time limit on the retention of skipped message keys in order to protect against an eavesdropper.

Compromising forward secrecy: The desktop client state information contains private pre-key material that will let the attacker break forward secrecy for the Signal account to which it is linked. Copying it results in the attack computer's Signal installation exactly mirroring the victim's installation. The attacker instance contains all historical messages from the victim installation and is able to receive and decrypt all future incoming messages as well as send encrypted messages that appear to be from the victim. The mobile device and legitimate desktop client do not have separate ratchets.

We tested de-linking the legitimate Signal desktop client from the phone, which initially made the cloned desktop unable to send or receive messages. Since the database decryption key was easily available, we loaded the signal app into an adb-emulated Android device running on Ubuntu. We used run time level hooks provided by Frida[29] to develop a `tls-interceptor` app. Then loaded the `tls interceptor` app into the Android device to capture the traffic. We were able to capture the primary device ID and password using the `tls-interceptor` and use them to update and reconnect the cloned (but de-linked) desktop client. We were then able to send and receive messages on the de-linked desktop client using the primary credentials. The details of the attacker machine do not appear to affect the success of this attack. This experiment was performed only for Signal as the database decryption keys were not so readily identifiable in the material cloned for other desktop clients.

Un-scoped threats, security and privacy consequences: TM_{Δ} for Signal desktop client reveals that it was not scoped for protection against *spoofing, repudiation, information disclosure, denial of service and elevation of privilege* with respect to STRIDE and *linkability and identifiability* for LINDDUN. As shown by the example tree in Figure 5, the messages that an adversary has access to can be used to infer the inter-relationship between the contacts of the victim and with the victim. This can lead to secondary and tertiary linkability between the victim and entities through their direct contacts.

†This design decision has been discussed on multiple occasions on the Signal Community Forum and GitHub issue tracker since 2017.

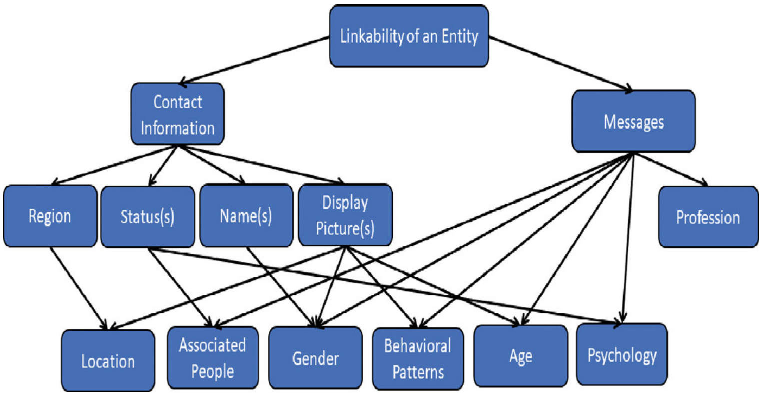

FIGURE 5 Linkability of an entity due to cloning of a device.

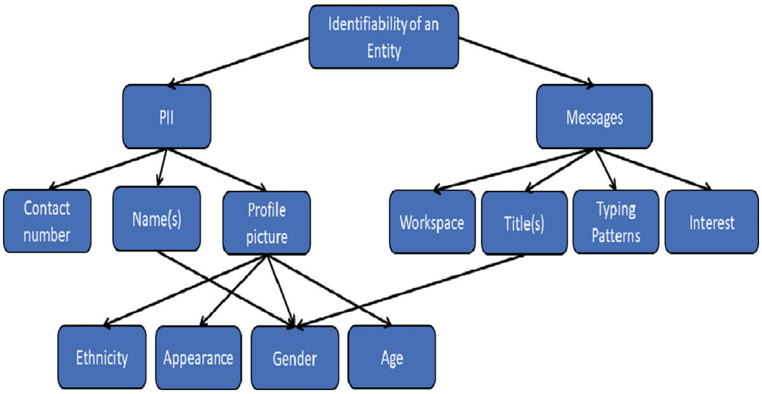

FIGURE 6 Identifiability of an entity due to cloning of a device.

Furthermore, as shown in Figure 6, a direct consequence of compromised DFD elements is that the attacker has further access to personally identifiable information and can identify the subjects and pair them with attributes.

> For the Signal messenger desktop, it is difficult to detect short-lived adversarial access and recover from a compromise. Access to the database decryption keys can render de-linking inconsequential. Adhering to the mobile-app threat model of external eavesdroppers betrays the reality that adversaries with transient access to linked devices might abuse this and escalate it to persistent access to the account.

5.2 | WhatsApp

WhatsApp desktop clients consider malicious entities other than an eavesdropper. However, our experiments show that their protection mechanisms were not very effective against such an adversary. An attacker with short-lived access can steal the credentials and communicate as the victim. However unlike Signal the key is not stored in plain text. This has a bearing on re-configuring a stolen desktop once it is de-linked from the primary device. Another improvement over Signal is that all companion devices have an expiry date. WhatsApp also alerts the primary device to new companion devices, as well as de-linking any existing companion devices in use. While spoofing is possible as in Signal, these design changes can limit the consequences.

Messages are still synchronized across devices, thus compromising confidentiality. An attacker is able to send and receive messages as the victim. This breaks forward secrecy and the recipients are not able to tell whether the messages are from an attacker or the victim.

Though there is a mitigation in the form of expiry dates, there are further privacy consequences for linkability and identifiability.

TM_{Δ} for the WhatsApp desktop client reveals that it was not scoped for protection against *spoofing, repudiation, information disclosure* with respect to STRIDE and *linkability and identifiability* for LINDDUN. However, the WhatsApp desktop client was scoped for protection against *elevation of privilege* and *denial of service* through short-lived adversarial access. The scoping for *spoofing* is a marginal improvement over the Signal desktop with the provision of expiry dates and alert messages for companion devices.

> Short-lived adversarial access does not reveal the database keys which, coupled with the alerts and expiry dates, protects the WhatsApp desktop user. The desktop client threat model considers that legitimate insiders can turn malicious but makes strong assumptions about users' ability to note and act on warnings and protect themselves.

5.3 | Viber

Our experiments show that Viber clones exited as soon as they were launched. This is because Viber explicitly pins the primary identity in the companion devices—configuring the companion device transfers the identity key pair to it. This is significant for both the mobile application and the desktop client. Any device willing to authenticate as a legitimate client needs to be explicitly authorized by the primary device. Though the state is stored in `~/Library/Application Support/Viber`, using this state requires the explicit transfer of the primary identity key by the legitimate owner.

When communicating entities are authenticated by each other without any interference from a man-in-the-middle, the trusted session is identified with a green lock. Any other session is marked with a red lock.

For Viber, TM_{Δ} reveals that it was scoped for protection against *spoofing, repudiation, information disclosure, denial of service and elevation of privilege* with respect to STRIDE and *linkability and identifiability* for LINDDUN.

> Viber appears well-scoped for the dynamic nature of the threat landscape by explicitly mandating companion devices through a transfer of the mobile device's primary ID. This eliminates the responsibility of users to detect cloning attacks and/or communications by the attacker and engage in the subsequent recovery.

5.4 | Wickr Me

Cloning attacks were not possible for Wickr Me even when the victim's state was set to "remember password." This implies that Wickr Me considered malicious participants beyond just eavesdroppers. Their mobile messaging application verifies the association of identity with its identity key pair and ephemeral key pairs. The association between the identity key pair and the identity is managed by the Wickr app and is pinned to the device, making it difficult for an attacker to authenticate as a victim. To protect from eavesdropping, Wickr Me encrypts server requests using a rotated shared secret using AES 256 in CFB mode which is tunnelled inside TLS. The security property of pinning the identity and key pair with the device also appears to be extended to the desktop client.

TM_{Δ} for Wickr Me reveals that it was scoped for protection against *spoofing, repudiation, information disclosure, denial of service and elevation of privilege* with respect to STRIDE and *linkability and identifiability* for LINDDUN.

> Wickr Me is distinct in the way it ties the device to an instance of companion device identity establishment—a cloned instance of a Wickr Me account will not be able to masquerade as the victim. Such implementations acknowledge the changing nature of the threats from actors with access to the devices, and the desirability of robust and verifiable bindings between cryptographic keys and real-world entities.

5.5 | Element

The desktop client considers malicious participants other than eavesdroppers to a greater extent than WhatsApp, but our cloning attacks still allow an attacker to find out who communicated with whom and when. While we moved the victim state to the attacker machine, the attacker was able to fire up the desktop client. The attacker was not able to send and receive messages and could not connect to the server, but they could see the user names of the entities with whom the victim communicated and when.

The mobile version of Element generates a secret key for every user of a container particular to a device, and we believe they extended this to their desktop version. The Matrix documentation states that they generate keys per device and not per user and that keys are never exported.[30] The keys cannot, therefore, be stolen and replicated in another device through a simple cloning attack. Thus, though cloning is possible in Element, it does not compromise forward and backward secrecy. However, the ability to see who sent messages to whom and when could still lead to linkability between the victim and their contacts.

In the case of Element, TM_{Δ} reveals that the desktop client was scoped for protection against *spoofing, repudiation, denial of service and elevation of privilege* with respect to STRIDE and *identifiability* for LINDDUN. Element desktop client also appears to be scoped for *information disclosure* for forward and backward secrecy but not for *linkability* with respect to LINDDUN.

> An attacker cannot compromise forward secrecy in Element through short-lived adversarial access. Element considers the threat of malicious insider access, and blocks access to message contents, but leaves users vulnerable to traffic analysis: a cloning attack still reveals the identity of communicating entities.

5.6 | Telegram

The mobile application has a cloud-based chat and an end-to-end secure chat, using MTProto 2.0. The protection primitives assume that a user is in control of the device. This assumption means that the user in control of the device can only authenticate with the long-term shared identity key. However, the desktop client's state information can be cloned through short-lived access and thus spoofed by an adversary. It is difficult for a recipient to distinguish between a legitimate sender and an attacker using a cloned account.

In the case of Telegram secret chats, message exchanges are not synchronized across legitimate and cloned devices. For secret chats, the client key pairs are replenished after every 100 messages or after being in use for more than a week. This is to prevent any compromise of forward secrecy. Participants in a secret chat can initiate key generation if and when they detect any compromise of their keys. However, attackers can also initiate secret chats with the contacts of the victim without the contacts or the victim being able to detect them. The disclosure of contact information leads to inferences about the contacts of the victims and leads to identifying sensitive information.

TM_{Δ} reveals that Telegram Desktop client was not scoped for *spoofing, repudiation, information disclosure* and *denial of service* with respect to STRIDE and *linkability and identifiability* for LINDDUN. The scoping for *spoofing* was a marginal improvement over Signal Desktop with the provision of multi-factor authentication and passwords (as well as the option to set an automatic logout after a period of time).

> Telegram desktop clients can be cloned and detection of a compromise is non-trivial for a victim. The ability to set expiry dates considers malicious insiders to some extent but leaves users to protect themselves. The adoption of the *eavesdropper-only* threat model in a context where access to account state information is easier, leaves users vulnerable to cloning attacks.

5.7 | Summary

Our comparative analysis of the desktop clients of the E2EE messaging applications has led to foundational considerations in security engineering. Some of the applications relied on *authentication* to secure their desktop clients; however, as our experiments show, that clearly has limits. When legitimate insiders become adversaries, Signal, WhatsApp and Telegram desktop clients are vulnerable to simple cloning attacks. Table 2 show the possibility of *spoofing, repudiation, information disclosure, denial of service, elevation of privilege, linkability* and *identifiability* on them. Element, WickrMe and Viber desktop clients, on the other hand, complement *authentication* with explicit handover of trust (from the mobile application to every companion device); consequently they are not vulnerable to simple cloning attacks. While *authentication* and *authorization* are traditional concerns in any privilege management infrastructure, our results show that they have limitations and need to be complemented with additional mechanisms for optimum protections. Therefore, proper scoping of threats is key to designing appropriate mechanisms.

6 | THREAT MODELING TO ALIGN TRUST BOUNDARIES WITH ADMINISTRATIVE BOUNDARIES

Distinct individual or collective entities function within a logical space (personal space, departments, or organizations). We refer to these as *administrative boundaries*. System designers evaluate threats within and across these administrative boundaries, specify security policies to counter the threats and provide appropriate controls to implement them. The artifacts protected by controls are placed within a *trust boundary* in order to mitigate particular threats. The key question is the extent to which trust and administrative boundaries should align. Delineating trust boundaries too restrictively will impose a heavy burden on legitimate users for example, frequent re-authentication. On the other hand, not scoping the trust boundary suitably can lead to bad actors within the administrative boundary being able to compromise security and privacy. And as discussed above, threat models can change—new threats can emerge to an existing application.[31,32] And additional features or extensions can change the administrative boundary leading to the emergence of new threats. This phenomenon is observable in our case study.

Mobile phones function within a culture where sharing of devices is not common. The threat model is focused more on external entities like eavesdroppers and middleboxes. System designers accordingly defined the security policies for their E2EE services to focus on the mobile device and implemented controls to protect against external threat actors. This is reflected in the trust boundary of the DFD in Figure 2 pertaining to TM_1, which contains the mobile phone.

Then when we see desktop clients added to messaging applications, TM_2 represents scenarios where there is a shift in the administrative boundary—external participants in official or domestic settings have easier access to the desktop clients, and normally legitimate insiders can turn malicious. Our investigation reveals distinct positions of the trust boundary depending on how companion devices can be fired up pertaining to their corresponding user account. These are captured in Figure 7. Signal messenger, WhatsApp and Telegram permit companion devices to be set up without the primary device; they can be cloned from an initial companion device with key material loaded by the primary device—allowing adversaries with short-lived adversarial access to a genuine companion device to clone it. Figure 7A

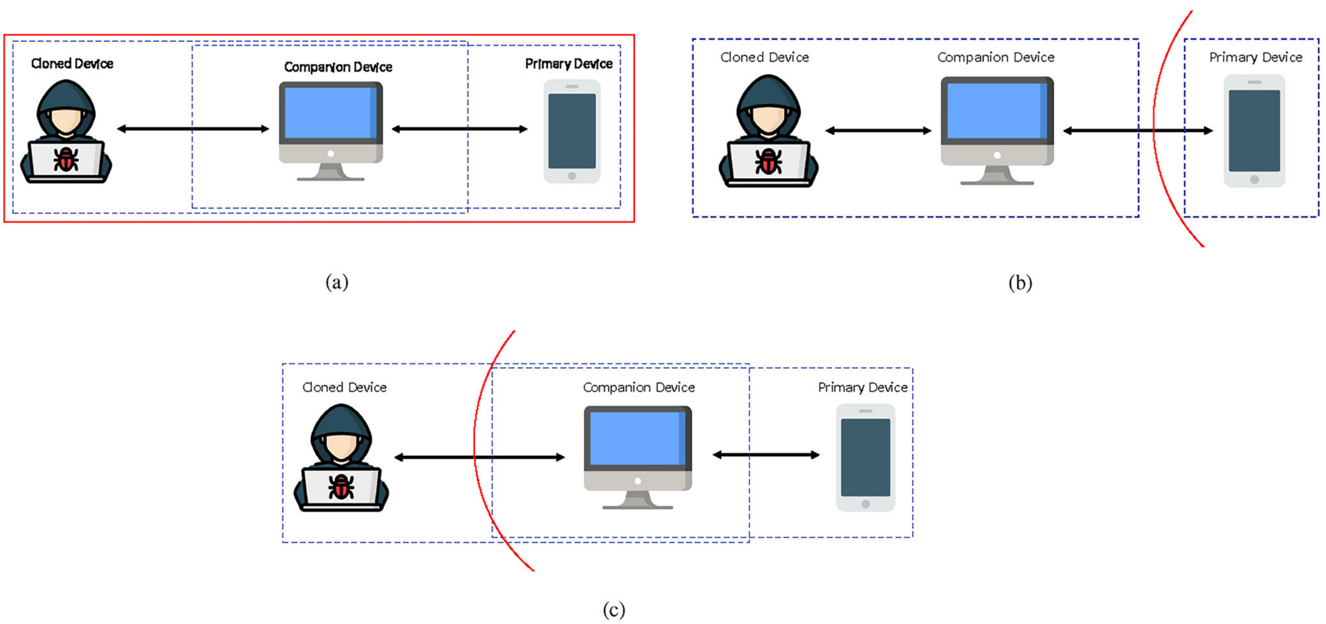

FIGURE 7 Placement of trust boundary (red line) with respect to administrative boundary (blue dotted line). (A) Mal-actors within the trust boundary. (B) Primary device only in trust boundary. (C) Mal-actors outside the trust boundary.

shows that this is because the trust boundary includes legitimate insiders who can turn malicious. Compare the scenario where the trust boundary contains only the mobile phone as in Figure 7B; this is secure but would require frequent authorization by the primary device.

On the other hand, the security controls for Viber, Element and Wickr Me require all companion devices to be fired up by the primary device. The trust boundary as depicted in Figure 7C includes the primary device and only those desktop clients explicitly fired by it. This excludes any potentially malicious insider with short-lived access to a client.

The question is how security engineering can rescope the trust boundary with changes in the administrative boundary as applications evolve over space and time. There have been several studies on the challenges of threat modeling in agile development environments[33] which acknowledge the emergence and disappearance of threats. A related study[34] reports operational challenges with threat modeling in agile environments and recommends including security experts on developer teams. The importance of experts is further supported by Assal et al. reported that the absence of security knowledge is a significant blocker in the adoption of secure coding practices by developer teams.[35]

Our investigation suggests that re-evaluation of trust boundaries in the light of TM_{Δ} need not be a highly resource-intensive task. For instance, it might build on lightweight interventions proposed in previous practice studies such as Weir et al.[9] They designed low-cost practical support for development teams—*Developer Security Essentials*. They propose that threat assessment could be low-cost and easy to implement with the aid of a suitable facilitator. A specific output of the threat assessment can be the evaluation of TM_{Δ} with respect to specific threat taxonomies.

6.1 | Scoping "often" to protect human rights

Appropriate scoping of the trust boundary with respect to the administrative boundary has implications for the human rights of direct users and implicated data subjects of E2EE messaging. Given the push in multiple jurisdictions for embedding content scanning tools in E2EE messaging applications, using arguments around protecting children and preventing terrorist radicalization, there is an incentive for some principals to obscure trust boundaries. Examples include police and intelligence agencies claiming that end-to-end encryption is not compromised by mandating government scanning of message content before it is encrypted and sent, or after it is decrypted and received. Clarity on trust boundaries and administrative boundaries may help legislators see through such arguments and come to more appropriate decisions.

We draw on the first public evaluation[36] of some prototype client-side scanning tools to highlight the importance of appropriate scoping of the trust boundary in technologies that claim to protect children without violating individual privacy.

Threats resulting from expanded memory scanning: A key consideration is whether content scanning tools can scan everything in the device where they are deployed. If they can, then mechanisms mandated to undermine the privacy offered by one messaging app will destroy the security and privacy of all other data on the device as well. The threat modeling approach described here will show that tools placed at the operating system layer, and with broad administrative privileges, situate themselves within the trust boundary of many security-critical artifacts.

The evaluation document[36] brings out the varying degrees to which pervasive memory scanning is performed by the candidate surveillance tools. Some allow pervasive scanning while others leave some discretion to the user. Even so, there will be serious issues around whether defaults are safe, and in line with citizens' expectations, and indeed with human rights law. Tools that undermine platform security rather than just circumvent the encryption mechanisms of an app with which they are bundled can lead to pervasive surveillance and abuse.

Threats due to embedding the tools within other applications: Content scanning tools can be placed within E2EE messaging applications or other applications, undermining their security and privacy controls. This has been flagged by the evaluation report.[36] For example, video conferencing and gaming platforms might be mandated to embed such tools. Threat modeling needs to consider cases where such users' security or privacy protection might be turned off remotely, whether coercively or surreptitiously or both, leaving the application re-purposed and the end user vulnerable. Appropriate scoping of the trust boundary will need to take into account the complex and diverse underlying memory protections offered by Windows, Android and iOS.[37] It is quite possible that an app with a mandated government backdoor might be a systemic surveillance and security threat on some platforms but not on others. This might lead to interesting policy externalities. For example, if a mandated backdoor in WhatsApp threatens Signal too on Windows devices but not on Apple devices, then users might either abandon WhatsApp or abandon Windows. Microsoft might then be motivated either to harden Windows or to lobby harder against the surveillance mandate.

6.2 | Summary

Expansion of feature sets should be accompanied with reimagining the administrative boundary and investigate if that leads to unauthorized exposure of security artifacts. Constraining the trust boundary might lead to usability issues but a broader trust boundary to accommodate usability needs might lead to compromise of security artifacts. The key is to place complementary mechanisms to compensate the change in administrative boundaries.

Threats resulting due to introduction of mechanisms for law enforcement needs should be scoped to prevent their misuse. Inappropriate placement of such tools might lead to system wide compromise beyond specific purposes they are meant for. When such tools are embedded within other tools, threat modeling should look if they are able to turn off user protections without the knowledge and consent of the end user.

When mandated tools to help law enforcement are embedded within other tools, they can lead to externalities. Configurations of the tool can compromise competing applications on one platform but not on another. This means users would either abandon the platform or the tool. Scoping of threats needs to be cognizant of such platform specific externalities that result due to economic reasons.

7 | DISCUSSION

Reconciliation of security requirements across components with shared state: The E2EE messaging applications were initially designed for mobile phones and the desktop clients followed later on—mobile phone application continued to be the root of trust for the desktop clients. A pertinent issue is if "trust" in the security of the mobile application is enough to trust the security of the desktop clients. On the other hand, can the compromise of the desktop client lower the security of the mobile application account as well?

Setting up a desktop client involves sharing credentials between the primary device and the secondary device—the credential is the shared state between the primary and companion device. As our analysis shows, for some of the desktop clients, the shared system state is open to compromise due to short-lived adversarial access.

Research in secure systems development has considered the question of when the security of the components is sufficient to trust the larger system (i.e., the composability problem).[38] Threat modeling of individual system components can help identify the shared state between them and analyze the consequences of such sharing. If sharing is inescapable then security (of the larger system) is perhaps incumbent on the administration of the shared components. In the context of security of open distributed processing, Bull et al. suggest independent administration of the components.[39] Gong et al. proposed a model for autonomous administration of shared state for secure inter-operation of systems.[40] Minimal sharing of state information has been a longstanding security principle, for example, *least common mechanism* in the Saltzer and Schroeder principles.[41]

Safe defaults: Systems are designed with clear delineation of the participants in the system. We learn from our investigation of the desktop clients of the messaging applications that some of them assume that these participants have fixed behavior which does not change across space and time. The manner in which applications respond to the change in the behavior of the participants determines the security of the applications. *Fail-safe defaults* have been a long-standing principle too.[41]

Developers, administrators and end users find it challenging to make fail-safe design and usage decisions.[42] There are some ways to facilitate such decisions: prompting and nudging developers when committing code may help.[43] There have been suggestions of contextualizing the principles for application developers; see for example Neumann's discussion of the limits of willpower, and what principles can achieve.[44] Overall, there appears to be a consensus that we still need more work on ways to make fail-safe decisions.[42]

8 | CONCLUSION

Software design can involve a tussle between functional requirements and their security implications, where functionality all too often wins. In an investigation of messaging clients, we have observed two contrasting ways in which they deal with the threats they face. One set of applications anticipated the possibility of client cloning and implemented systems-level mitigation strategies, while another put the onus squarely on the user to prevent device compromise in the first place. While most of the desktop clients are based on similar cryptographic primitives, their diversity is a reflection of their varying perceptions of likely attackers. Designers with a more realistic appreciation of the threat produced more robust systems. The larger lesson here is system maintainers have to keep their threat model up to date, especially if their product is successful and acquires millions of users, or they will be left behind and a mismatch between their threat model and reality may leave users exposed to attacks.

We argue for nuanced and in-depth modeling of attackers in appropriate contexts as integral to the software development lifecycle. Application features evolve—existing features are deprecated or updated and new ones added—and so must threat models. The model of the attacker cannot be independent of the model, and indeed of measurement, of system users. There are users of shared devices and or managed devices. The administrative boundaries expand with the participation of occasional adversarial users such as border and customs officials, nosy bosses and abusive intimate partners. The growing diversity of entities within the administrative domain exposes users to previously unscoped threats.

Our suggestion for application designers is to not only do threat modeling little and often but also to pay close attention to TM_Δ when doing so. Understanding how a threat model evolves over space and time is key to rescoping trust boundaries to cope. It is also critical to evaluate whether misalignment of administrative and trust boundaries leads to incorrect assumptions about the threats posed by insiders, whom we model here as those located within an administrative boundary but who ought not to be deemed trustworthy. Usability also matters: security controls can be too strict leading to circumvention or to a product being abandoned. Avoiding poor usability outcomes, and thus poor commercial outcomes means keeping trust boundaries and administrative boundaries realistically aligned.

Future threat modeling work should consider how development teams can use tools and concepts such as TM_Δ to map the gap between different types of boundary and maintain situational awareness of threats that arise because of features added elsewhere—or indeed of changes in assumptions about the broader threat environment.

Appropriate scoping of threats is perhaps even more critical for cyber-physical systems. While we discuss security and privacy violations in this article, the risks are exacerbated when failure leads to safety violations. For example, software defects in medical devices lead to deaths and injuries,[45] with prior research showing that a significant number of deaths were caused due to usability variations between devices and comparable to deaths due to road traffic accidents.[46] Expansion of usability features can be accompanied with scoping appropriate risks that evolve through discussions with operators who use those machines. On a general note appropriate scoping of threats should accompany changes in feature sets and usability variations.

AUTHOR CONTRIBUTIONS

Partha Das Chowdhury was co-lead in the experiments, led writing the paper with substantial contribution. Maria Sameen was STRIDE threat modelling, LINDDUN mapping and building the attack trees, diagrams and contributed to writing. Jenny Blessing was co-lead in the experiments, conceptualization, contributed to developing our work in the light of related work, methodology, contributed to writing the paper. Nicholas Boucher was co-lead in the experiments, conceptualization, methodology, review and editing the paper. Joseph Gardiner was co-lead in the experiments, conceptualization, methodology, review and editing the paper. Tom Burrows was co-lead in the experiments, developed the tls-Intercept used in this work. Ross Anderson was mentor for the entire project, co-lead in the experiments, framed the arguments in discussion and scientific relevance of this work, continuously reviewed and edited the paper. Awais Rashid was mentor during the experiments, framed the narrative of the paper through the introduction and other sections and contributed significantly towards reviewing and editing the paper.

ACKNOWLEDGMENTS

This work is supported by REPHRAIN: Research Centre on Privacy, Harm Reduction and Adversarial Influence Online (EPSRC Grant: EP/V011189/1).

CONFLICT OF INTEREST STATEMENT

To the best of our knowledge we do not have any conflict of interest with respect to this work.

DATA AVAILABILITY STATEMENT

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

ORCID

Partha Das Chowdhury https://orcid.org/0000-0002-5367-6659

Maria Sameen https://orcid.org/0000-0002-6086-8974
Jenny Blessing https://orcid.org/0009-0007-7470-6435
Nicholas Boucher https://orcid.org/0000-0002-5674-3730
Joseph Gardiner https://orcid.org/0000-0003-4748-4228
Awais Rashid https://orcid.org/0000-0002-0109-1341

REFERENCES

1. Howard M, Lipner S. *The Security Development Lifecycle*. Vol 8. Microsoft Press; 2006.
2. The OWASP Foundation. OWASP secure coding practices quick reference guide. https://owasp.org/www-pdf-archive/OWASP_SCP_Quick_Reference_Guide_v2.pdf
3. SAFECode. Practical security stories and security tasks for agile development environments. http://safecode.org/wp-content/uploads/2018/01/SAFECode_Agile_Dev_Security0712.pdf
4. Microsoft. Secure DevOps. https://www.microsoft.com/en-us/securityengineering/devsecops
5. Deng M, Wuyts K, Scandariato R, Preneel B, Joosen W. A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. *Requir Eng*. 2011;16(1):3-32.
6. Gumbley J. A guide to threat modelling for developers: secure software design, little and often. https://martinfowler.com/articles/agile-threat-modelling.html
7. Spring JM. An analysis of how many undiscovered vulnerabilities remain in information systems. *Comput Secur*. 2023;131:103191.
8. Bernsmed K, Cruzes DS, Jaatun MG, Iovan M. Adopting threat modelling in agile software development projects. *J Syst Softw*. 2022;183:111090.
9. Weir C, Becker I, Noble J, Blair L, Sasse A, Rashid A. Interventions for software security: creating a lightweight program of assurance techniques for developers. *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE; 2019:41-50.
10. Weir C, Migues S, Ware M, Williams L. Infiltrating security into development: exploring the world's largest software security study. *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM; 2021:1326-1336.
11. BBC. Moxie Marlinspike leaves encrypted-messaging app signal. https://www.bbc.co.uk/news/technology-59937614
12. BBC. Facebook, Whatsapp and Instagram back after outage. https://www.bbc.co.uk/news/technology-58793174
13. Chatterjee R, Doerfler P, Orgad H, et al. The spyware used in intimate partner violence. *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP 2018), May 21-23, 2018, San Francisco, CA*. IEEE Computer Society; 2018:441-458.
14. Tseng E, Bellini R, McDonald N, et al. The tools and tactics used in intimate partner surveillance: an analysis of online infidelity forums. *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*. USENIX Association; 2020:1893-1909.
15. Jakobsen JB, Orlandi C. *A Practical Cryptanalysis of the Telegram Messaging Protocol*. Aarhus University; 2015.
16. Albrecht MR, Mareková L, Paterson KG, Stepanovs I. Four attacks and a proof for telegram. *43rd IEEE Symposium on Security and Privacy (IEEE S&P 2022)*. IEEE; 2022:87-106.
17. Albrecht MR, Celi S, Dowling B, Jones D. Practically-exploitable cryptographic vulnerabilities in matrix. https://nebuchadnezzar-megolm.github.io/static/paper.pdf
18. Cremers C, Fairoze J, Kiesl B, Naska A. Clone detection in secure messaging: improving post-compromise security in practice. *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. ACM; 2020:1481-1495.
19. Ventura V. in(Secure) messaging apps—how side-channel attacks can compromise privacy in WhatsApp, Telegram, and signal. https://blog.talosintelligence.com/2018/12/secureim.html
20. Marlinspike M. The double ratchet algorithm. https://signal.org/docs/specifications/doubleratchet/#introduction
21. Howell C, Leavy T, Alwen J. Wickr messaging protocol technical paper. https://wickr.com/wp-content/uploads/2019/12/WhitePaper_WickrMessagingProtocol.pdf
22. Diffie W, Hellman M. New direction in cryptography. *IEEE Trans Inf Theory*. 1976;22:472-492.
23. Williams L. Secure software lifecycle. *The Cyber Security Body of Knowledge v1.1.0, 2021. KA Version 1.0.2*. University of Bristol; 2021. https://www.cybok.org/
24. Shevchenko N, Chick TA, O'Riordan P, Scanlon TP, Woody C. *Threat Modeling: A Summary of Available Methods*. Technical Report. Carnegie Mellon University Software Engineering Institute; 2018.
25. WhatsApp. WhatsApp encryption overview. Technical white paper. https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf
26. Element. Matrix specification. https://element.io/enterprise/end-to-end-encryption
27. Telegram. MTProto mobile protocol. https://core.telegram.org/mtproto/description
28. Viber. Viber encryption overview. https://www.viber.com/app/uploads/viber-encryption-overview.pdf
29. FRIDA. Dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers. https://frida.re/docs/home/
30. The Matrix.org Foundation. Client-server API (unstable). 2021 https://spec.matrix.org/unstable/client-server-api/
31. Omoronyia I, Cavallaro L, Salehie M, Pasquale L, Nuseibeh B. Engineering adaptive privacy: on the role of privacy awareness requirements. *2013 35th International Conference on Software Engineering (ICSE)*. IEEE; 2013:632-641.
32. Sgandurra D, Lupu E. Evolution of attacks, threat models, and solutions for virtualized systems. *ACM Comput Surv*. 2016;48(3):1-38.

33. Galvez R, Gurses S. The odyssey: modeling privacy threats in a brave new world. *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE; 2018:87-94.
34. Cruzes DS, Jaatun MG, Bernsmed K, Tøndel IA. Challenges and experiences with applying microsoft threat modeling in agile development projects. *2018 25th Australasian Software Engineering Conference (ASWEC)*. IEEE; 2018:111-120.
35. Assal H, Chiasson S. Security in the software development lifecycle. *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*. USENIX Association; 2018:281-296.
36. Peersman C, Llanos JT, May-Chahal C, McConville R, Chowdhury PD, De Cristofaro E. *Towards a Framework for Evaluating CSAM Prevention and Detection Tools in the Context of End-to-End-Encryption Environments: A Case Study*. REPHRAIN; 2023.
37. Abelson H, Anderson R, Bellovin SM, et al. Bugs in our pockets: the risks of client-side scanning. arXiv preprint arXiv:2110.07450, 2021.
38. Ross R, McEvilley M, Oren J. *Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems*. Technical Report. National Institute of Standards and Technology; 2016.
39. Bull JA, Gong L, Sollins KR. Towards security in an open systems federation. *European Symposium on Research in Computer Security*. Springer; 1992:3-20.
40. Gong L, Qian X. Computational issues in secure interoperation. *IEEE Trans Softw Eng*. 1996;22(1):43-52.
41. Saltzer JH, Schroeder MD. The protection of information in computer systems. *Proc IEEE*. 1975;63(9):1278-1308.
42. Jaeger T. Toward fail safety for security decisions. *IEEE Secur Priv*. 2021;19(06):4-7.
43. Chen H, Ciborowska A, Damevski K. Using automated prompts for student reflection on computer security concepts. *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. ACM; 2019:506-512.
44. Neumann PG. Fundamental trustworthiness principles. In: Shrobe H, Shrier D, Pentland A, eds. *New Solutions for Cybersecurity*. MIT Press/Connection Science; 2018.
45. Leverett E, Clayton R, Anderson R. Standardisation and certification of the 'Internet of Things'. *Proceedings of WEIS*. University of Cambridge; 2017.
46. Thimbleby H. Improving safety in medical devices and systems. *2013 IEEE International Conference on Healthcare Informatics*. IEEE; 2013:1-13.

How to cite this article: Das Chowdhury P, Sameen M, Blessing J, et al. Threat models over space and time: A case study of end-to-end-encrypted messaging applications. *Softw: Pract Exper*. 2024;1-20. doi: 10.1002/spe.3341