REPHRAIN

Protecting citizens online

Threat Models over Space and Time: A Case Study of E2EE Messaging Applications

Partha Das Chowdhury, University of Bristol

Maria Sameen, University of Bristol

Jenny Blessing, University of Cambridge

Nicholas Boucher, University of Cambridge

Joseph Gardiner, University of Bristol

Tom Burrows, University of Cambridge

Ross Anderson, University of Cambridge/University of Edinburgh

Awais Rashid, University of Bristol

UK Research and Innovation · University of Bristol · UCL · The University of Edinburgh · King's College London · University of Bath

August 2023

Threat Models over Space and Time: A Case Study of E2EE Messaging Applications

Partha Das Chowdhury*, Maria Sameen*, Jenny Blessing†, Nicholas Boucher†, Joseph Gardiner*, Tom Burrows†, Ross Anderson†‡, Awais Rashid*

*University of Bristol, UK {partha.daschowdhury, maria.sameen, joe.gardiner, awais.rashid}@bristol.ac.uk

†University of Cambridge, UK. {jenny.blessing, nicholas.boucher, ross.anderson}@cl.cam.ac.uk, tom@tpmb.uk

‡University of Edinburgh, UK ross.j.anderson@ed.ac.uk

Abstract—**Threat modelling is foundational to secure systems engineering and should be done in consideration of the context within which systems operate. On the other hand, the continuous evolution of both the technical sophistication of threats and the system attack surface is an inescapable reality. In this work, we explore the extent to which real-world systems engineering reflects the changing threat context. To this end 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 with respect to two popular threat elicitation frameworks, STRIDE and LINDDUN. The results demonstrate that system designers need to both recognise the threats in the evolving context within which systems operate and, more importantly, to mitigate them by rescoping trust boundaries in a manner that those within the administrative boundary cannot violate security and privacy properties. Such a nuanced understanding of trust boundary scopes and their relationship with administrative boundaries allows for better administration of shared components, including securing them with safe defaults.**

1. Introduction

Threat modeling has become an integral part of secure software development. Threat modeling is also a recommended best practice by OWASP [1] and within Agile [2] and DevOps processes [3]. Researchers have developed similar frameworks to systematically analyze threats to user privacy when developing software applications [4]. However, threat modeling should not be a one-off activity. As threats evolve and new attacks come to light, developers must continuously reassess applications against these emerging threats. Furthermore, adding new features to an application creates new information flows, which in turn can cause trust boundaries to shift (e.g., due to including additional hardware or software components or third party services). The recommended best practice is to do *threat modeling "little and often"* [5].

In this paper, we analyze whether the threat models that underpin security and privacy aspects of applications evolve through space (as new features are added) and time

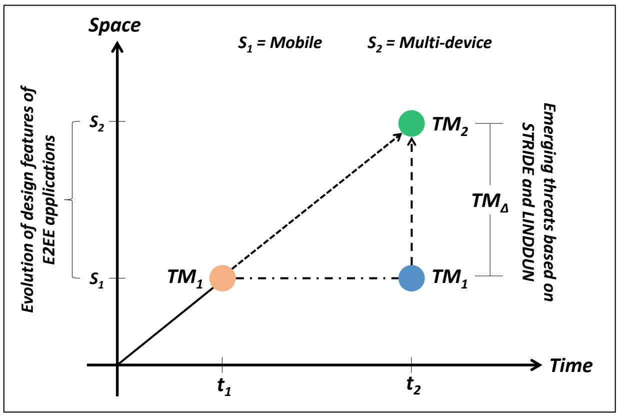


Figure 1: Evolution of Threat Models

(as understanding of threats changes). We use the case study of end-to-end-encrypted (E2EE) mobile messaging applications (such as Signal [1] and WhatsApp) that have found widespread uptake and adoption amongst users aiming to protect the privacy of their communications and mitigate large-scale surveillance [6], [7]. Most of these messaging platforms have since launched their desktop clients to make it easier for users to communicate across multiple devices.

At the same time, 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 utilize monitoring technology or shared devices as a way to monitor and exert control over the victim [8], [9]. In this scenario, the threat actor is not remote and has direct physical access to devices. Furthermore, for an abuse victim, the persistence of the 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 such as **official searches** by border & customs, **shared devices** in households, **corporate managed devices** so on and so forth.

Figure 1 represents a conceptualization of the impor-

1. We refer to the Signal messenger application here, rather than the Signal protocol, and will continue to distinguish between the two throughout the paper.

tance 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, i.e., a mobile app client with a remote attacker. We then develop threat models (TM_2) encompassing the expanded feature space and understanding of threats with respect to security (using the STRIDE threat modeling approach [10]) and privacy (using the LINDDUN threat modeling approach [4]). Using an experimental test setup, we then simulate adversarial short-lived access to the desktop clients of each of the six applications. We use any resultant compromises to derive 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 privacy leakages: *linkability* of information and *identifiability* of the communicating parties. Our analysis also highlights that recognition of the change in threat context is useful but not enough in itself unless backed by appropriate countermeasures.

2. Background - E2EE Messaging Applications & Threat Frameworks

We summarize the key security properties of these applications and their desktop clients in Table 1 followed by the threat modelling frameworks STRIDE [10] and LINDDUN [4].

2.1. E2EE Messaging Applications

1) Every installation is tied to a particular user identity – namely, identity key IK. This identity is then used as the root of trust to securely communicate with other participants through the service and to configure additional devices for the same account.
2) There are ephemeral asymmetric key pairs known as pre-keys, which are used to encrypt messages between communicating entities. The public component of the pre-keys are communicated to the server for other communicating entities – this communication is signed by the private component of the long term identity key pair. The assumption is that only the owner of the account has access to the private part of IK and thus signed it.
3) Signal, WhatsApp, Element and Viber broadly adhere to the Signal protocol's Double Ratchet algorithm [11], as shown in Table 1. Wickr Me extends the foundation of identity as the root of trust their adoption of the Wickr secure messaging protocol [12]. Telegram is a MTProto 2.0 based messenger.

The desktop clients for E2EE apps also share some common characteristics:

- Every user needs to have an account with their primary device which is the mobile application and the legitimate user is in control and possession of the primary device.
- After standard installation of the desktop client for any of the messaging application, when launched, they generate their own identity key pair. This identity key pair is distinct from the primary device identity key pair.
- The primary device scans the identity information of the desktop client and they authenticate each other. The desktop client proves legitimate ownership of the identity key to the primary device. The primary device then communicates to the corresponding application server that the desktop client is trusted and can communicate as the primary device.
- The primary device retains a list of the linked companion devices.

A common security assumption across all the E2EE applications we investigate is that account holders will be able to keep their identity key and its corresponding secret away from attackers.

2.2. Threat Modeling

STRIDE [10] is a widely known and utilized security threat modeling approach. The system is modeled in the form of Data flow Diagrams (DFDs) that capture the key processes, data stores and data flows between them. Trust boundaries delineate which partitions of the system are assumed to be free from adversarial interference. Data flows across trust boundaries and individual components (processes and data stores) are then evaluated for their susceptibility to six key threats: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege. This susceptibility is derived through a negation of key security properties, respectively, authentication, integrity, non-repudiation, confidentiality, availability and authorization.

LINDDUN [4] follows a similar approach to STRIDE but focuses on threats to privacy. The system is modeled using DFDs. Then individual DFD elements and data flows are evaluated for potential privacy threats: Linkability, Identifiability, Non-repudiation, Detectability, information Disclosure, content Unawareness and Noncompliance. The threats are elicited through negation of key privacy properties, respectively, unlinkability, anonymity/pseudonymity, plausible deniability, undetectability/unobservability, confidentiality, content awareness and policy/consent compliance.

3. Methodology

Choice of E2EE Applications. We selected applications that are generally widely used and are diverse in how they establish trust between primary and companion devices. The applications studied can be divided into two broad categories: those that rely in some capacity on

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.

TABLE 1: Properties of Popular Messaging Applications

the Signal protocol, and those that do not. For Signal-based apps, we study two implementations of the Signal protocol, Signal's own messaging app and WhatsApp, where Signal's app is open-source while WhatsApp's implementation is closed-source. This allows us to gain a comparative understanding of the trust establishment (of companion devices by the primary device) between Signal and WhatsApp (as both utilize the Signal E2EE protocol as their foundation). As we note later in our findings, there are noticeable differences between their respective implementations. While not relying directly on the Signal protocol, Viber and Element both make use of the Double Ratchet algorithm in their implementations. 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 also noteworthy due to its decentralized nature (i.e., it does not rely on a central communications server). We investigate if this decentralization has any bearing on trust establishment of the companion devices by the primary devices.

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

Creation of DFDs. We created DFDs (using Microsoft's Threat Modeling Tool) for each app before and after addition of the Desktop client by studying the security properties in their documentation [6], [12]–[16] and through our experiments (see Section 3.1). For example, WhatsApp documentation explicitly states (page 25) [13]

> "*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 our parent company Facebook, can access the content in between.*"

Figure 2 depicts the DFD at time t_1 and space S_1 (cf. Figure 1). 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 by the apps.

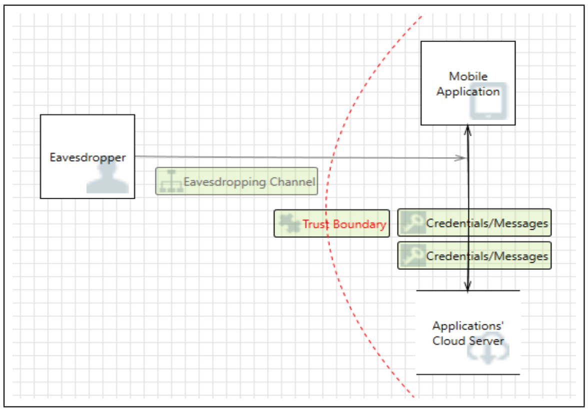


Figure 2: DFD for Signal, WhatsApp, Element, Wickr Me, Viber, and Telegram mobile applications.

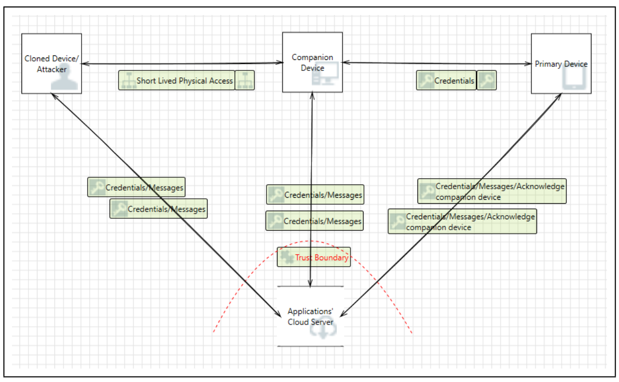


Figure 3: DFD for Signal, WhatsApp, and Telegram desktop applications.

3.1. 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 exper-

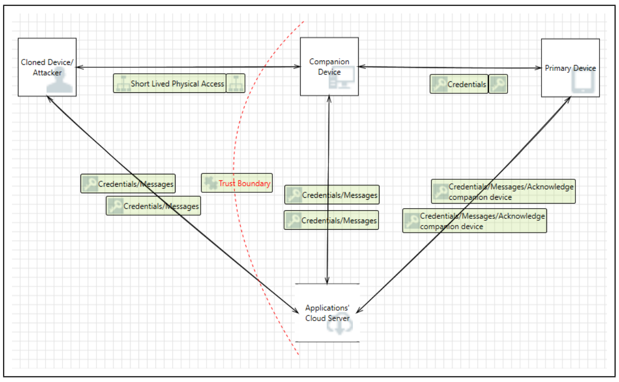


Figure 4: DFD for Element, Wickr Me and Viber desktop applications.

iments. 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 16GB 3733 MHz LPDDR4X memory.
2) **Launching the legitimate desktop client**. We started a legitimate desktop version of the application through the required setup mechanism, for example, by scanning a `QR code`.
3) **Launching the attacker's desktop client**. We performed a standard installation of the desktop client in 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 machine. Our goal was to evaluate if these systems have protections against simple cloning attacks so we copied state information from $\sim$`/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 specifications.

3.2. Testing for threats

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 and placed it in the attacker's machine.
- **Tampering**. Our focus was on the end points 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 other legitimate parties. This was repeated between the communicating entities to understand if the recipients observe any difference while communicating with the victim and the attacker.
- **Information Disclosure**. We used the cloned desktop across space and time to understand the implications on forward and backward secrecy.
- **Denial of Service**. We tested if 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.

LINDDUN. We focused on identifiability and linkability as information disclosure and non-repudiation (vice versa repudiation) were already evaluated as part of our STRIDE analysis. Since our focus was on end points, we did not perform analysis for detectability. Non-compliance and unawareness are out of scope for us as the registration processes for the messaging applications were based on primary device credentials without any room for opting out.

- **Linkability**. – The various artefacts from a victim were checked if they link potential entities connected to the victim.
- **Identifiability**. – We checked if those artefacts 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.

4. Findings

We subject the desktop client to short-lived adversarial access (TM_2) to elicit the threats based on the tests particular to elements of the threat as per 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.

4.1. Signal Messenger

Signal messenger assumes that only an eavesdropper can be or can attempt to be the mal-actor and implements its authentication and key sharing mechanisms accordingly. For any other mal-actors the expectation is to replace the device and/or account. This was perhaps appropriate for implementing the security property of authentication because the secrets never left the user's device into the communication network. However, our experiments show that such assumptions cannot sustain when the potential mal-actors reside within the trust boundary and adversarial short-lived access can go undetected.

An attacker can simply replace the configuration files of a standard Signal desktop installation with the versions stolen from a victim's machine. The specifics of the attacker machine do not influence the success of the

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

TABLE 2: ($TM_{\triangle}$) based on STRIDE and LINDDUN threat models. In this table, (-) indicates not being tested; (×) indicates attack not possible; and (✓) indicates attack is possible.

attack. For Signal mobile application, upon new installation (either due to compromised device or for other reasons), the identity keys of the user change along with the pre-keys. This will be reflected for all the contacts of a particular account. A comparison between the legitimate desktop version and the cloned desktop showed the same keys against the sequence number of the pre-keys. This shows that the DH ratchet is not effective at rendering the cloned version obsolete after the existing key material is exhausted. One caveat is that the attacker's Signal desktop instance will work with delays or messages will be dropped when the victim's mirrored desktop installation is actively online. Our observation was that it was dependent on the session established either with the victim or the attacker. The desktop client state information contains private pre-key material that will let the attacker break forward secrecy for the Signal account it represents.

$TM_{\triangle}$ for Signal desktop client captured in Figure 6 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.

Adhering to the (mobile app) threat model of external eavesdroppers betrays the reality where other participants with access to the devices (for legitimate reasons or due to proximity) might misuse such access.

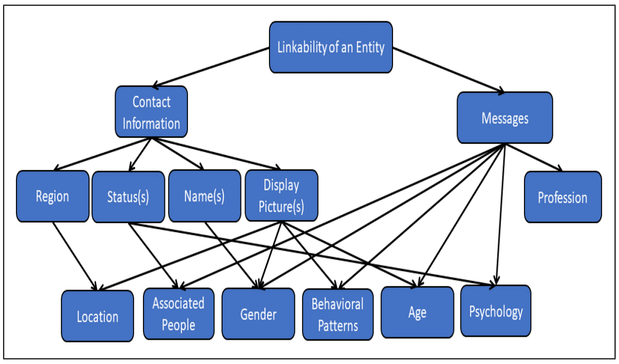


Figure 5: Linkability of an Entity due to cloning of a device

4.2. WhatsApp

WhatsApp desktop clients consider malicious entities other than an eavesdropper. However, our experiments show that their protection mechanisms were not commensurate against such an adversary. An attacker with short-lived access can steal the credentials and communicate as

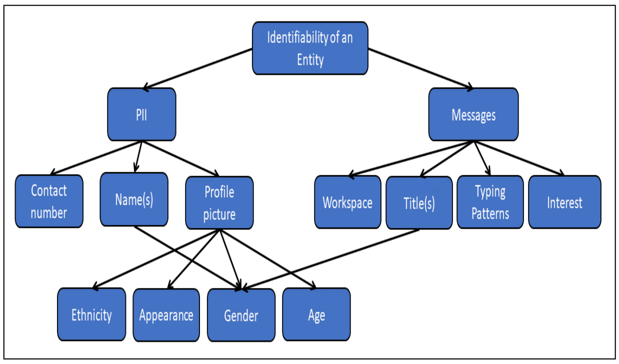


Figure 6: Identifiability of an Entity due to cloning of a device

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 implementation improvement over Signal is that all companion devices have an expiry date. WhatsApp explicitly alerts the primary device for new companion devices as well as existing companion devices in use. While spoofing is possible similar to Signal, there are implementation improvements which can limit the consequences.

The messages are 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 distinguish if the messages are from an attacker or the victim. Though there is a limit in the form of expiry dates, yet there are privacy consequences namely linkability and identifiability.

$TM_{\triangle}$ for 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, 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 Signal desktop with the provision of expiry dates and alert messages for companion devices.

The desktop client threat model though considers that legitimate insiders can turn malicious but makes strong assumptions on users' ability to note and act on warnings and protect themselves.

4.3. 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` yet using the state requires to explicit transfer of the primary identity key by the legitimate owner. When communicating entities are authenticated by each other without any interference of a man-in-the-middle, the trusted session is identified with a green lock. Any session otherwise is indicated 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 primary ID.

4.4. Wickr Me

The cloning attacks were not possible for Wickr Me even when the victim's state was set to remember password. This means that Wickr Me considered malicious participants beyond just eavesdroppers. Their mobile messaging application verifies the association of an identity with their identity key pair and ephemeral key pairs. The association between identity key pair and the identity is managed by the Wickr app and is pinned with the device, making it difficult for an attacker to authenticate as a victim. To protect from an eavesdropper Wickr Me encrypts server requests using a rotated shared secret using AES 256 in CFB mode which is tunneled 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 acknowledges the changing nature of the threats from actors with access to the devices and also opens the possibility to integrate robust and verifiable bindings between cryptographic keys and real world entities.

4.5. Element

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

The mobile version of Element generates a secret key pertaining to every user for a container particular to a device which we believe they extended for their desktop version. Matrix documentation states that they generate keys per device and not per user and the keys are never exported from the device [17]. The keys are not part of the state that can be stolen and replicated in another device through simple cloning attack. Though cloning is possible in Element, it does not compromise forward and backward secrecy. The ability to see who sent messages to whom and when could lead to linkability between the victim and their contacts. However, the message contents are not exposed due to cloning so an attacker cannot gather identifiable attributes of the communicating entities.

In 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.

> A cloning attack against Element reveals only the identity of the communicating entities but lack of access to the keys prevents an adversary from reading the messages.

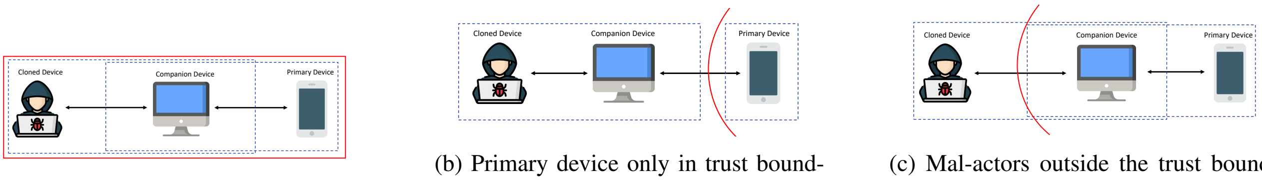
4.6. Telegram

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. 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 contacts of the victim without the contacts or the victim being able to detect it. 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).

> The adoption of the *eavesdropper-only* threat model in a context where access to the account state information is easier, leaves users vulnerable to cloning attacks.



(a) Mal-actors within the trust boundary

(b) Primary device only in trust boundary

(c) Mal-actors outside the trust boundary

Figure 7: Placement of trust boundary (red line) with respect to administrative boundary (blue dotted line).

5. Threat modelling to align trust boundaries with administrative boundaries

Distinct individual or collective entities function within an *administrative boundary* — logical space (personal space, departments or organizations). System designers evaluate threats within and across these administrative boundaries, specify security policies to counter the threats and provision appropriate security controls to implement these policies. The artefacts protected by security controls are placed within a *trust boundary* in order to mitigate particular threats. The key question that arises is the extent to which trust and administrative boundaries should align.

Mobile phones function within a administrative boundary where sharing of devices is not a widely prevalent reality. The threat model is focussed more on entities external to the administrative boundary like eavesdroppers and attackers in the middle. System designers accordingly defined the security policies and mechanisms for their mobile E2EE services to protect against such threat actors. This is reflected in the trust boundary of the DFD in Figure 2 pertaining to TM_1, which contains the mobile phone. In case of the desktop clients of the messaging applications, TM_2 represents scenarios where there is a shift in the administrative boundary — external participants in official environment or for statutory reasons or domestic settings have easier access to the devices with desktop clients. Threats can arise from otherwise legitimate insiders turning malicious.

Our investigation captured in Figure 7 shows the distinct positions of the trust boundary depending on how companion devices can be fired up pertaining to their corresponding user account. The security controls of Signal messenger, WhatsApp and Telegram do not prevent companion devices being set up without the primary device. There can be companion devices cloned from an initial companion device fired by the primary device — allowing mal-actors with short-lived adversarial access to do so. As we can observe from Figure 7a, this is because the trust boundary includes legitimate insiders who can turn malicious. For a comparative understanding we refer to a scenario where the trust boundary situates with only the mobile phone as in Figure 7b. Such a model would result in usability load requiring frequent authorization by the primary device to the secondary device. On the other hand, the security controls for Viber, Element and Wickr Me require that all companion devices are explicitly fired by the primary device. The trust boundary as depicted in Figure 7c includes the primary device and only those desktop clients explicitly fired by the primary device. This excludes any (potential malicious) insider who has short-lived access to a desktop client.

Our investigation shows that re-evaluation of trust boundaries in the light of TM_Δ needn't be a highly resource intensive task. For instance, such re-evaluation can build on lightweight interventions proposed in previous practice studies such as Weir et al. [18]. They designed low cost practical support for development teams — *Developer Security Essentials*. Among the proposed interventions, threat assessment is one of the key yet low cost and easy to implement intervention with the aid of a facilitator. A specific output of the threat assessment (at various stages) can be the evaluation of TM_Δ with respect to specific threat taxonomies. This in turn will determine the security policies and controls to rescope the trust boundaries of systems against any possible change to their administrative boundaries.

6. Discussion

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. Can the compromise of the desktop client, on the other hand, lower the security of the mobile application account as well? 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) [19].

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 behavior of the participants determine the security of the applications. *Fail safe defaults* has been a long standing principle too [20] — participants with access to devices should not be able to use the access maliciously. However, there remains a need for further work to find systematic ways to make fail-safety decisions [21] as well as exploring new mechanisms for the suitability of *deny access* as a *fail safe default* [22].

7. Related Work

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 MT-

Proto protocol and its implementation [23], [24]. Most recently, the Matrix protocol (which Element uses) suffered several severe vulnerabilities in which a malicious actor was able to break confidentiality of communications using a compromised Matrix server [25]. Cremers et al. [26] 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.

In this paper, 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. Evolution of applications is a reality as is addition of new features. However, rescoping of the trust boundary as both the application and the understanding of threats evolves is critical to mitigating against emerging threats. Our analysis of the delineation of trust and administrative boundaries provides a basis for such rescoping and better administration of shared components, including securing them with safe defaults.

8. Conclusion

Software is an intersection between its functional requirements and their security implications, where functionalities take a precedence. In our investigation, we observe two contrasting ways in which desktop clients deal with the threats they face. One set of applications anticipate the possibility of client cloning and implement various systems-level mitigation strategies, while a second set puts the onus squarely on the user to prevent device compromise in the first place. An additional observation from our experiments is that the addition of a less secure desktop client can substantially lower the security of the mobile client. While most of the desktop clients are based on the same or similar cryptographic primitives, their manifest diversity is a reflection of their varying perceptions of an attacker. The larger lesson here is that securing evolving threats with assumptions made in a different context will result in a mismatch leaving users exposed to attacks. We make a case for nuanced and in-depth modelling of an attacker 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 do threat models. Our key take away 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.

References

[1] The OWASP Foundation. OWASP secure coding practices quick reference guide. https://owasp.org/www-pdf-archive/OWASP_SCP_Quick_Reference_Guide_v2.pdf.

[2] SAFECode. Practical security stories and security tasks for agile development environments. http://safecode.org/wp-content/uploads/2018/01/SAFECode_Agile_Dev_Security0712.pdf.

[3] Microsoft. Secure DevOps. https://www.microsoft.com/en-us/securityengineering/devsecops.

[4] M. Deng, K. Wuyts, R. Scandariato, B. Preneel, and W. Joosen, "A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements," *Requirements Engineering*, vol. 16, no. 1, pp. 3–32, 2011.

[5] J. Gumbley. A guide to threat modelling for developers: Secure software design, little and often. https://martinfowler.com/articles/agile-threat-modelling.html.

[6] BBC. Moxie Marlinspike leaves encrypted-messaging app Signal. https://www.bbc.co.uk/news/technology-59937614.

[7] ——. Facebook, Whatsapp and Instagram back after outage. https://www.bbc.co.uk/news/technology-58793174.

[8] R. Chatterjee, P. Doerfler, H. Orgad, S. Havron, J. Palmer, D. Freed, K. Levy, N. Dell, D. McCoy, and T. Ristenpart, "The spyware used in intimate partner violence," in *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*. IEEE Computer Society, 2018, pp. 441–458.

[9] E. Tseng, R. Bellini, N. McDonald, M. Danos, R. Greenstadt, D. McCoy, N. Dell, and T. Ristenpart, "The tools and tactics used in intimate partner surveillance: An analysis of online infidelity forums," in *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*. USENIX Association, 2020, pp. 1893–1909.

[10] M. Howard and S. Lipner, *The security development lifecycle*. Microsoft Press Redmond, 2006, vol. 8.

[11] Trevor Perrin (editor), Moxie Marlinspike. The Double Ratchet Algorithm. https://signal.org/docs/specifications/doubleratchet/#introduction.

[12] Chris Howell, Tom Leavy, Joël Alwen. Wickr Messaging Protocol Technical Paper. https://wickr.com/wp-content/uploads/2019/12/WhitePaper_WickrMessagingProtocol.pdf.

[13] WhatsApp. WhatsApp Encryption Overview Technical white paper. https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf.

[14] Element. Matrix Specification. https://element.io/enterprise/end-to-end-encryption.

[15] Telegram. MTProto Mobile Protocol. https://core.telegram.org/mtproto/description.

[16] Viber. Viber Encryption Overview. https://www.viber.com/app/uploads/viber-encryption-overview.pdf.

[17] The Matrix.org Foundation. "Client-Server API (unstable), May 2021". https://spec.matrix.org/unstable/client- server- api/.

[18] C. Weir, I. Becker, J. Noble, L. Blair, A. Sasse, and A. Rashid, "Interventions for software security: Creating a lightweight program of assurance techniques for developers," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2019, pp. 41–50.

[19] R. Ross, M. McEvilley, and J. Oren, "Systems security engineering: Considerations for a multidisciplinary approach in the engineering of trustworthy secure systems," National Institute of Standards and Technology, Tech. Rep., 2016.

[20] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975.

[21] T. Jaeger, "Toward fail safety for security decisions," *IEEE Security & Privacy*, vol. 19, no. 06, pp. 4–7, 2021.

[22] F. Massacci, "Is" deny access" a valid" fail-safe default" principle for building security in cyberphysical systems?" *IEEE Security & Privacy*, vol. 17, no. 5, pp. 90–93, 2019.

[23] J. B. Jakobsen and C. Orlandi, "A practical cryptanalysis of the telegram messaging protocol," *Aarhus university*, 2015.

[24] M. R. Albrecht, L. Mareková, K. G. Paterson, and I. Stepanovs, "Four attacks and a proof for telegram," in *43rd IEEE Symposium on Security and Privacy (IEEE S&P 2022)*, 2022.

[25] Martin R. Albrecht, Sofia Celi, Benjamin Dowling, Daniel Jones. Practically-exploitable Cryptographic Vulnerabilities in Matrix. https://nebuchadnezzar-megolm.github.io/static/paper.pdf.

[26] C. Cremers, J. Fairoze, B. Kiesl, and A. Naska, "Clone detection in secure messaging: improving post-compromise security in practice," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1481–1495.