WhatsApp with privacy? Privacy issues with IM E2EE in the Multi-device setting

Tal A. Be'ery, *Zengo*

https://www.usenix.org/conference/woot24/presentation/beery

This paper is included in the Proceedings of the 18th USENIX WOOT Conference on Offensive Technologies.

August 12–13, 2024 • Philadelphia, PA, USA

ISBN 978-1-939133-43-4

Open access to the Proceedings of the 18th USENIX WOOT Conference on Offensive Technologies is sponsored by USENIX.

WhatsApp with privacy? Privacy issues with IM E2EE in the Multi-device setting

Tal A. Be'ery, *Zengo*

Abstract

We recently discovered a privacy issue with Meta's WhatsApp, the world's most popular Instant Messaging (IM) application. Meta's WhatsApp suffers from a privacy issue that leaks the victims' device setup information (mobile device + up to 4 linked devices) to any user, even if blocked and not in contacts. Monitoring this information over time allows potential attackers to gather actionable intelligence about victims and their device changes (device replaced/ added /removed). Additionally, message recipients can associate the message with the specific sender device that sent it. The root cause for these issues stems from Signal's multi device protocol architecture, the Sesame protocol, and as a result these issues are not limited to Meta's WhatsApp only but probably relevant to most IM solutions, including the privacy-oriented Signal Messenger.

1. Introduction

End-to-End Encryption (E2EE) is a type of messaging that keeps messages private from everyone, including the messaging service. When E2EE is used, a message only appears in decrypted form for the person sending the message and the person receiving the message. The sender is one "end" of the conversation, and the recipient is the other "end"; hence the name "end-to-end.".
Originally, most Instant Messaging (IM) apps did not support E2EE. However, as the importance and criticality of IM security had raised, E2EE became the security standard for modern communication and supported by modern IM apps.

Another aspect of IM communications that evolved over time is its multi-device support. Traditionally, Instant Messaging (IM) apps were bounded to a single device. However, as IM have gained popularity and became an important and even critical medium for communications, users wanted to have access to their IM conversations from every computing device they own. As a result, modern IM providers support the multi-device setting.

While each of these individual features (E2EE and multi-device) is critical for modern IM apps, supporting both simultaneously can lead to some security and privacy tradeoffs, as current E2EE solution expose some public cryptographic information about each of the devices, by thus compromising their users' privacy.

Contributions: Our main contributions are the following:

- We show the privacy and integrity implications of current popular multi-device solutions in IM apps.
- We demonstrate how attackers can easily subvert the WhatsApp client to obtain the victims' multi-device setup information.
- We suggest some practical measures to limit the exposure of such privacy leaks.

Overview: This paper is organized as follows: Section 2 provides a brick-and-mortar analogy to IM E2EE, Section 3 presents the Signal protocol and highlights the privacy issues in the multi-device setting, Section 4 shows how such privacy leaking attacks can be easily mounted by attackers against WhatsApp currently the world's most popular IM service, section 5 considers possible solutions. We conclude in Section 6.

2. Background

To better understand E2EE and its threat model we can use the postal service analogy:
Prior to E2EE, senders sent their letter in an envelope, but the envelope was not sealed. As part of its service, the post office opens the envelope and then puts it in another envelope and delivers it to the intended recipient.

This scheme has many advantages:

- Thanks to envelopes, eavesdroppers cannot see the contents of the letters.
- Thanks to the post office buffering, users do not need to meet to converse, but rather do so indirectly. This not only allows asynchronous conversations but also can protect user anonymity. Receivers can disclose only their nicknames to senders, and have the post service resolve from nicknames to true names and addresses. In fact, there is a privacy tradeoff between service and the conversation counterparty: If the conversing parties are directly connected, then the service is not exposed to the contents of the conversation, however the parties may uncover more metadata about each other and be able to break the "rules" of the protocol as the service is not there to enforce them. Generally speaking, it makes sense to assume that the service provider is more trustworthy than some counterparties that might be malicious.
- The post office can scan the contents of envelopes to make sure they do not contain bad content: Bombs, terror group messaging or pedophile photos.
- If letters are intended for multiple addresses (groups or a user with multiple houses) the post office can simply copy the message and send it to all addresses.

However, this scheme has a major drawback: Postal service employees are exposed to the contents of the letters and can leak them. The practical reasons for such leakage can vary: The service may act in negligence and mishandle user data, sell the data to advertisers for financial gain, be hacked by attackers, fail to restrict rogue employee access to private customer data, or even be served with a subpoena by the government.

To address this issue, E2EE was introduced. With E2EE, users send their message in locked boxes within the envelopes. Users provide their locks to the service when they join, but keep the keys themselves. When senders want to send a letter to a recipient they get the relevant padlock from the service and send their letter in a locked box within the envelope. As before, the post office opens the envelope and then puts it in another envelope and delivers it to the intended recipient. However, due to the locked box, the postal service personnel can no longer see the contents of the letter.

While E2EE indeed protects message content from the prying eyes of the service operator, it should be noted that:

- Even with E2EE, users must place some trust in the service provider, as the storing and forwarding messages, even encrypted, exposes metadata. Whether it's conversation related (counterparties, number of messages, length of messages, timing) or operational (online status, devices used, IP addresses which may have geo-location information).
- The newly added E2EE lock creates a new identifier for the user. When users lose their key, they must issue a new lock for the service. Aware attackers might leverage this information to deduce something changed on the user side.
- To make sure the E2EE lock is indeed of the intended user and not maliciously replaced by the service or a "Monster-in-the-Middle" (MITM) attacker, the sender must verify the lock's genuinity with the receiver using another independent channel. This requirement not only hinders the user experience but also jeopardizes the privacy of the users as they need to connect via additional service with additional identifiers.

But even with E2EE, users were still concerned: What happens if attackers break into their homes? Surely the system cannot prevent attackers from unlocking boxes and reading letters while they are still there and can use the keys, but we want to make sure that this privacy breach is limited to the exact period of the breach. Namely:

- Perfect Forward Secrecy (PFS): Attackers cannot open locked boxes that were locked before they broke into their victims' homes.
- Post Compromise Security (PCS): Attackers cannot open locked boxes that were locked after they left their victims' homes.

To achieve these properties, keys must be updated for every message, such that in case of compromise, the compromised keys are only useful for that message only. To do so, the two parties within the conversation are sending information to update the next locks and keys within their conversation.

It should be noted that while the first scenario of the pre-E2EE postal service privacy leak might be relevant at a large scale, for example to read the conversation of many users for serving ads or for mass surveillance, the case of post E2EE breaking into victims' homes does not scale well and mostly relevant to a small portion of the population consisting of highly targeted individuals. Since the contents of the messages themselves cannot be protected during the time of the attackers breaking in, the scenario for which PCS and PFS are relevant is only when attackers break into the victims' homes along with compromise of the service to get some of the victims' locked message boxes. Having such two successful independent attacks is a much less likely scenario than each of these attacks on their own.

3. The Signal protocol: From postal service analogy to real world crypto

3.1. The basic Signal protocol

WhatsApp is using the Signal protocol to implement E2EE's "postal service locked boxes" with public key cryptography. Users create their private and public key pair on their device when they join the IM service, and provide their public keys (possibly along with additional auxiliary data) to the IM service, which maintains the directory of the user's public keys. When parties wish to converse, the IM server provides them with their counterparty's public keys. It should be noted, as discussed above, that the newly added E2EE public key creates a new identifier for the user. When users lose their device, they must issue a new pair of keys for the service. Aware attackers might leverage this information to infer changes on the user side and leverage them to facilitate attacks.

Leveraging both parties' public keys, the parties can securely create a shared secret using the X3DH protocol, an extended version of the Diffie-Hellman protocol. This shared secret is then used to derive keys to encrypt the messages between the parties. While this in of itself might be sufficient to fulfill E2EE's promise, in order to fulfill the advanced properties of E2EE, namely the aforementioned Perfect Forward Secrecy (PFS) and Post Compromise Security (PCS), more is needed.

- Perfect Forward Secrecy (PFS): Attackers cannot read messages that were encrypted before they took over the victims' device and app.
- Post Compromise Security (PCS): Attackers cannot read messages that were encrypted after they were removed from the victims' device and app.

As discussed above, to limit breached key exposure and achieve PCS and PFS, a new key for each message needs to be created. To do so the Signal protocol introduced the "Double Ratchet" algorithm. As its name suggests, the solution consists of two "ratchets" preventing attackers compromising a key to "move it forward" to read future encrypted messages, or "backwards" to read past encrypted messages:

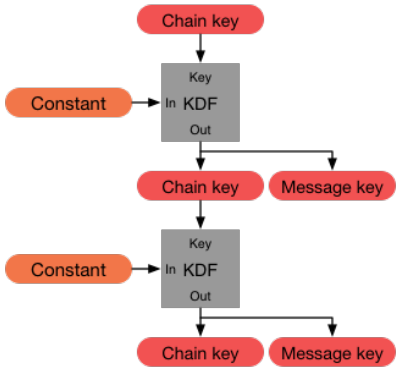

Figure 1 The Symmetric ratchet (source: signal.org)

The Symmetric ratchet (Fig 1): Ensures PFS, as it uses a one-way Key Derivation Function (KDF) to prevent attackers from calculating past keys from current keys.

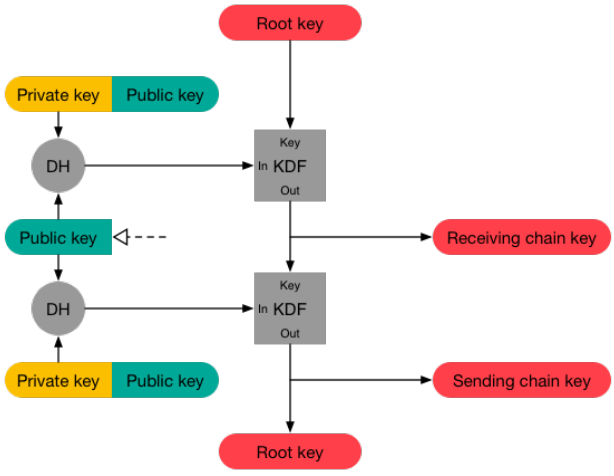

Figure 2 The Asymmetric ratchet (source: signal.org)

The Asymmetric ratchet (fig 2): This ratchet (sometimes called the "Diffie-Hellman/DH ratchet") ensures PCS as it utilizes the entropy coming from the uncompromised other party to generate new keys.

Combining the symmetric and asymmetric ratchets together gives the Double Ratchet: When a message is sent or received, a symmetric-key ratchet step is applied to the sending or receiving chain to derive the message key.

When a new ratchet public key is updated via a received message, a DH ratchet step is performed prior to the symmetric-key ratchet to replace the chain keys.

3.2. Extending E2EE to the Multi Device setting: Existing solutions

As discussed above, in the pre E2EE era, the multi-device support requirements were trivial to solve. Since the IM server had access to the contents of the message, senders could just send their message once to the server, totally unaware of the receiver's device setup and the IM server would handle its distribution to all of the receiver's devices and sender's other devices (so that their history would be up to date). However once E2EE is applied, the IM server cannot read the contents of the message and thus can no longer distribute them to all of the devices.

IM providers needed to address E2EE in the multi-device setting while still maintaining PCS and PFS requirements. Extending PFS and PCS definitions for the multi-device setting is quite natural:

- Perfect Forward Secrecy (PFS): Attackers cannot read messages that were encrypted before they took over the victim's app on one device.
- Post Compromise Security (PCS): Attackers cannot read messages that were encrypted after they took over the victim's app on one device and were removed from it.

There are two simple solutions to do so:

The "Leader" based solution: One of the user's devices serves as the leader and the E2EE conversation happens between the parties leaders, in the same manner as if both users had a single device. The leaders then distribute the messages to their other devices, using E2EE between Leader and Devices. In the WhatsApp mobile based IM case, it would be natural to appoint the mobile device which was associated with the phone number that created the account as "leader" (or "primary device" in WhatsApp lingo).

This solution was applied by WhatsApp until mid-2021. However, the solution suffers from an obvious centralization drawback: When the leader device is offline, none of the other devices can communicate.

The Multiplication solution: In this solution, all user device public keys become public, compared to a single public key per user in the single device setting. The sender's sending device creates an E2EE channel with each of the receiver devices, as if they were different users and uses these channels to send E2EE messages in the same manner of the single device setting. The sender device also creates such channels with all other sender's devices and uses them to securely E2EE update other senders devices with the sent messages.

This Multiplication solution was selected by Signal's Sesame protocol to support the multi-device setting, and later adopted by WhatsApp, where it serves as its current solution for the multi-device setting.

WhatsApp's white-paper states: "In order for WhatsApp users to communicate with each other securely and privately, the sender client establishes a pairwise encrypted session with each of the recipient's devices. Additionally, the sender client establishes a pairwise encrypted session with all other devices associated with the sender account. Once these pairwise encrypted sessions have been established, clients do not need to rebuild new sessions with these devices unless the session state is lost, which can be caused by an event such as an app reinstall or device change. WhatsApp uses this "client-fanout" approach for transmitting messages to multiple devices, where the WhatsApp client transmits a single message N number of times to N number of different devices. Each message is individually encrypted using the established pairwise encryption session with each device."

It should be noted that WhatsApp still uses the leader concept for managing the life cycle of additional devices (or "companion device" in WhatsApp lingo). i.e. adding and removing other devices is done via the "leader" device.

While this Multiplication solution solves its predecessor's centralization problem, it also multiplies the E2EE privacy issue. The multiplication solution exposes all of the user's device setups and allows aware attackers to leverage this information to infer changes in the user's devices and use it to facilitate their attacks. For example, attackers can learn without interacting with their targets, that they added a device to their setup and thus represent an opportunity to attack it. Additionally, the receiver knows which of the sender devices' sent to it and can infer on the sender's real-world information, such as the physical location of the user (e.g. "my spouse is near their desktop right now").

Besides device information leakage issues, the Multiplication solution potentially allows attackers to pinpoint their attack to a specific device. Since the sender creates independent channels with the receiver devices, it can send a malicious message to a single receiver's device to exploit a vulnerability specific to it, e.g. mobile vs. desktop exploit, with no impact and thus detection opportunities for defenders on other devices.

Additionally, a rogue sender can create an incoherent world view between the victim's different devices, by sending a different message to each of them. This incoherent world view can give way to all kinds of social engineering attacks and generally undermine the credibility of the IM app messages history as a source of truth.

The threat of device hostile takeover is very much within the IM's E2EE threat model, as shown by the existence of the PFS and PCS requirements. Since device takeover is within the threat model, the privacy of users' devices exposed by the Multiplication solution which allows attackers to gather information for such takeover should be addressed too.

4. Attacking WhatsApp E2EE Solution

Meta's WhatsApp is the most popular messaging app in the world, with over five billion downloads and 2.4 billion active users.

One way for attackers to obtain WhatsApp users' device information is by leveraging WhatsApp web client. (It should be noted that this issue is not specific to the web version and is relevant for all WhatsApp client's platforms. However, the Web environment is the easiest way to demonstrate this issue as it does not require jail breaking or other additional hacking method to access the app's internal databases.) This client is using the browser's local storage to store the devices' identity key.

The browser's developer tools provide an easy way to view the contents of this table ("Signal-storage.identity-store") as depicted in Figure 3.

Figure 3 The identity store table: contacts' devices and Keys.

This table is storing all of the user's contacts and their corresponding identity keys. Primary devices are identified by the phone number and the '.0' suffix, while companion devices have a ':<n>.0' suffix (e.g. ":16.0").

By sampling a few instances, we had verified that this table's data indeed corresponds to the actual user devices.

For example, user X (in figure 4) has 1 primary device and 3 companion devices:

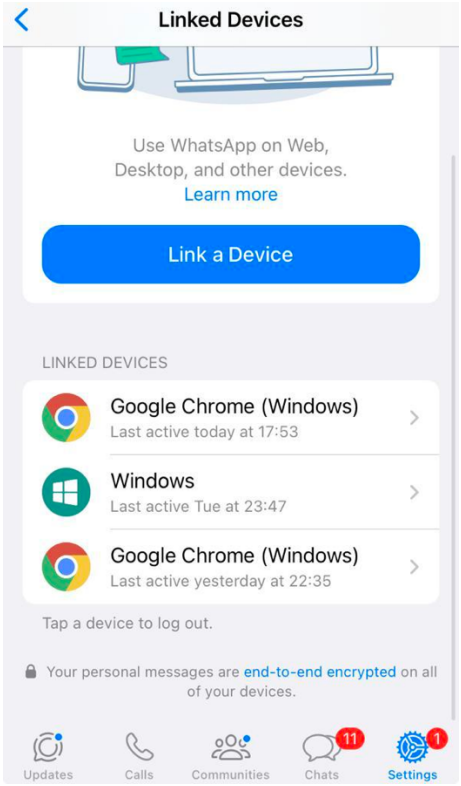

Figure 4 WhatsApp's linked devices screen of user X

User X's corresponding entries in the table matched this information as shown in Figure 5.

0	"9	4.0"	▸ {identifier:
1	"9	4:25.0"	▸ {identifier:
2	"9	4:26.0"	▸ {identifier:
3	"9	4:29.0"	▸ {identifier:

Figure 5 User X's corresponding entries in the identity store table

We had verified that such information is present even when the sender is not part of the receiver contact list and without actually sending messages to the receiver. Blocking the sender on the receiver side does not prevent it from getting device identity information.

We had responsibly disclosed our findings to Meta's bug-bounty program on January 9th 2024 but got politely rejected two days later, mainly because this is not an implementation bug but the way the protocol works by design.

Summing up, in order to obtain its victims' WhatsApp devices information, attackers need to:

- Know their victims' phone number.
- Add victims as contacts, no need to actually send a message to them.
- Use whatsApp web client and monitor the identity-store table for information and changes.

5. Possible solutions

5.1. "Lockdown mode" to limit non-contacts access

This optional Lockdown mode will enable users to limit messages' reception to ones sent by their contacts only. Consequently, only the users' contacts will need and be able to view their device information.

While it does not fully prevent the privacy issue it presents a dramatic improvement compared to the current situation in which any user, including blocked users, can view that information.

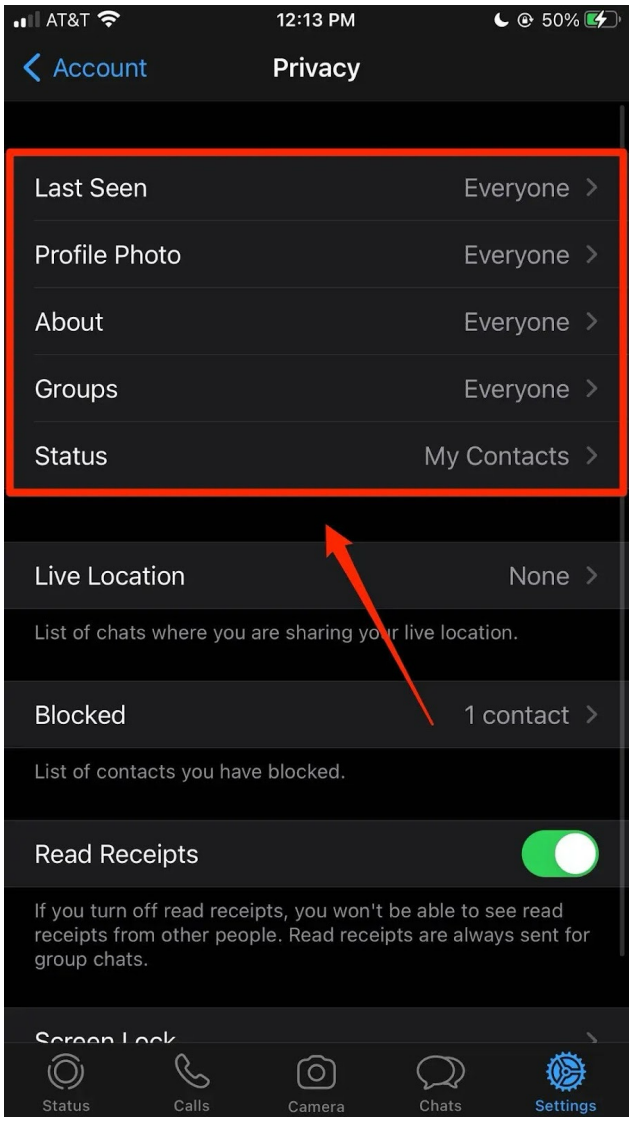

Figure 6 WhatsApp's privacy settings

This Lockdown mode can be beneficial to security and privacy aware users across the board and not just for this multi-device privacy issue, as it would protect them from receiving all kinds of malicious messages from non-contacts, which may include 0-days exploits, social engineering and phishing or even just spammy messages. The notion of limiting certain types of information to contacts only is

already present in WhatsApp as shown in Figure 6 and therefore already understood by its users.

5.2. Cryptographic solutions

To completely solve this issue a design change must be done, and the burden of distributing the messages needs to be removed from senders and placed on the receivers' instead.

As a result, the senders are only aware of a single recipient key, regardless of the number of the recipient's devices and are not aware of all recipients' devices and keys and cannot monitor changes to this setup.

A few researchers tried to suggest such solutions in the past, including a 2019 paper named "Multi-Device for Signal" that considers the multi-device scenario for the Signal protocol, which is used by WhatsApp (and others) and explicitly addresses and solves its privacy issues. It will be worthy to try and actually implement it or similar solution in popular IM E2EE solutions.

6. Conclusions

In this paper, we present the security and privacy tradeoffs of IM apps supporting both E2EE and multi-device. We demonstrate how attackers can easily subvert the WhatsApp client to obtain the victims' multi-device setup information and suggest some practical measures to limit the exposure of such privacy leaks.