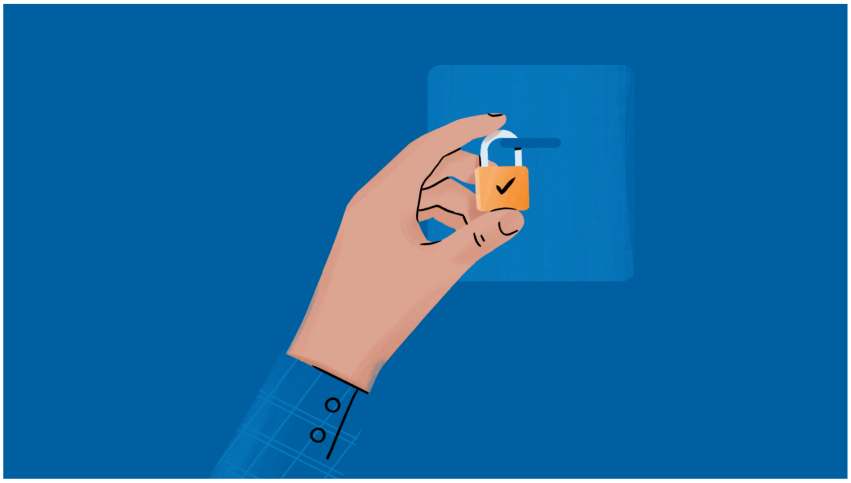
POSTED ON DECEMBER 6, 2023 TO SECURITY

Building end-to-end security for Messenger



By Jon Millican, Reed Riley

- We are beginning to upgrade people's personal conversations on Messenger to use end-to-end encryption (E2EE) by default.
- Meta is publishing two technical white papers on end-to-end encryption:
  - Our Messenger end-to-end encryption whitepaper describes the core cryptographic protocol for transmitting messages between clients.
  - The Labyrinth encrypted storage protocol whitepaper explains our protocol for end-to-end encrypting stored messaging history between devices on a user's account.

Today, we're announcing that we've begun to upgrade people's personal conversations on Messenger to use E2EE by default. Our aim is to ensure that everyone's personal messages on Messenger can only be accessed by the sender and the intended recipients, and that everyone can be sure the messages they receive are from an authentic sender.

This is the most significant milestone yet for this project, which began in earnest after Mark Zuckerberg outlined his vision for it in 2019. Bringing E2EE to Messenger has been a complex process, with every feature and product goal revealing further challenges that required careful consideration.

Enabling E2EE on Messenger meant fundamentally rebuilding many aspects of the application protocols to improve privacy, security, and safety while simultaneously maintaining the features that have made Messenger so popular.

Why we're bringing E2EE to Messenger

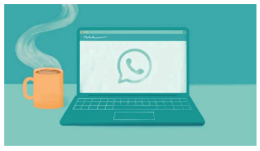
Messenger first built end-to-end encrypted chats in 2016 as a feature called Secret Conversations. Since then, we've learned a great deal in regards to rolling out E2EE for a wider user base. For example, we recently published an updated white paper, "Meta's Approach to Safer Private Messaging on Messenger and Instagram Direct Messaging," that sets out the industry-leading safety systems and tools available on Messenger.

End-to-end encryption isn't about the technology at its core. It's about protecting people's communications, so they can feel safe expressing themselves with their friends and loved ones. To achieve this, we typically focus on two aims:

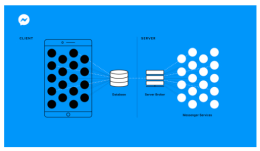
1. Only the sender and recipients of an E2EE message can see its contents.
2. Nobody (not even Meta) should be able to forge messages to appear to have been

Related Posts

Jul 28, 2022
Five security principles for billions of messages across Meta's apps



Mar 10, 2022
Code Verify: An open source browser extension for verifying code authenticity on the web



Mar 02, 2020
Project LightSpeed: Rewriting the Messenger codebase for a faster, smaller, and simpler messaging app

Related Positions

Security Analyst - Bug Bounty
TEL AVIV, ISRAEL

Security Analyst - Bug Bounty
LONDON, UK

GRC Integrity Program Manager
BELLEVUE, US

GRC Integrity Program Manager
AUSTIN, US

GRC Integrity Program Manager
MENLO PARK, US

See All Jobs

sent from someone they weren't.

In other words, the aim is that only you and the people you're corresponding with can read your messages – not even the app's provider (in this case, Meta) could interfere with their contents – and you can be confident in who sent the messages.

Understanding these goals

These two aims are broad. However, when we reflect on our approach to addressing them, they end up breaking down into eight overlapping concepts that we believe achieve a cohesive approach to meaningful E2EE:

1. Confidentiality in transit

Message contents are authentically and securely transmitted between your devices and those of the people you're talking to. This is, perhaps, the primary goal of E2EE, and is where much E2EE research and design work is targeted, such as the Signal protocol we use in our products (such as WhatsApp, Messenger, and Instagram Direct), or the IETF's Messaging Layer Security protocol, which we helped to design and was recently standardized.

2. Confidentiality in storage

Typically, E2EE messaging services rely on local storage and encryption keys to secure encrypted messages. Messenger, however, has a long history of storing people's messages for them so that they can access them whenever they need without having to store them locally. That's why we've designed a server-based solution where encrypted messages can be stored on Meta's servers while only being readable using encryption keys under the user's control.

3. Control over endpoints

For something to be "end-to-end encrypted," it is necessary to have a notion of what the "ends" are. For an E2EE messaging app this means that users should have the ability to verify and manage their set of endpoint devices that are receiving their messages, as well as visibility into when this set of devices changes.

4. Private feature designs

Product features in an E2EE setting typically need to be designed to function in a device-to-device manner, without ever relying on a third party having access to message content. This was a significant effort for Messenger, as much of its functionality has historically relied on server-side processing, with certain features difficult or impossible to exactly match with message content being limited to the devices.

5. Logging limitations

Maintaining the confidentiality of message content extends to avoiding accidentally leaking it back to us in telemetry. In a product of Messenger's scale, complexity, and iteration speed, this creates particular challenges as telemetry is vital in ensuring that the product is working well for people, and in debugging when things go wrong.

6. Application security

It's a common saying that, "You can't have privacy without security," and this is absolutely true in the end-to-end encrypted domain. Security is important for any consumer product, but E2EE exacerbates the challenges in two important ways: it reduces the provider's ability to protect the user from attacks, and, in fact, it expands the threat model to include the service provider itself. Our security team is keenly aware of these challenges and works closely with product teams to secure design and implementation of E2EE functionality. For example, we've been working to improve the memory safety of our apps; and our E2EE surfaces are covered by our bug bounty program.

7. Being deliberate about what's being protected

E2EE protects message content. However, this is a complex term to define, and, while certain things are relatively clear – such as the strings contained in a text message, or a photograph sent from your camera roll – in a sufficiently complex messaging application, it turns out there's a surprisingly large grey area. Our focus is on determining the appropriate boundaries, ensuring that we remain true to our commitments, setting the correct user expectations, and avoiding creating meaningful privacy risks, while still ensuring that the product retains its usefulness to our users.

8. Third-party scrutiny

E2EE implies confidentiality even if the provider wants to access the contents of a communication. We aim for this to be verifiable externally, and, to this end, have published

communication. We aim for this to be verifiable externally, and, to this end, have published two white papers to provide transparency into our operations. We describe the properties of some features in our Help Center, and we encourage submissions to our bug bounty program. Throughout the project, we have consulted with a diverse range of external parties to ensure that we're making the right set of tradeoffs. To improve people's ability to scrutinize us, we also support the Code Verify browser extension for our web-based end-to-end encrypted messaging, to give security researchers greater confidence that the code version that they are assessing is being used globally.

High-level approach

With all of this in mind, our high-level approach was to build off of Meta's prior learnings in E2EE, from both WhatsApp and Messenger's Secret Conversations, and then to iterate on our most challenging problems.

Working from the baseline of these two approaches, we then had to address a series of significant technical challenges, including:

1. **Multi-device capability**: Messenger's model of multi-device reflects the Facebook network, which allows people to authenticate on new devices with a username and password, in order to send and receive messages. Since WhatsApp's multi-device capability relies on a single primary device that must cryptographically authenticate companion devices, we adopted the Secret Conversations model of multi-device, while ensuring that it functions well for all of our users.
2. **Feature support**: Messenger has a number of messaging features that either don't exist in WhatsApp, or function differently. Some of these just had to be rebuilt from scratch, while others required deploying new applied privacy technology. For example, we used OHAI and Anonymous Credentials to support searches of Facebook's first-party sticker library, without revealing to us who is sending them.
3. **Message history**: Messenger has always allowed clients to operate off of a small stored local cache, relying on a server-side database for their message history. Neither WhatsApp nor Secret Conversations operated in this manner, and we didn't want all users to have to rely on a device-side storage system. Instead, we designed an entirely new encrypted storage system called Labyrinth, with ciphertexts uploaded to our servers and loaded on-demand by clients, while operating in a multi-device manner and supporting key rotation when clients are removed.
4. **Web support**: We needed to support E2EE within our existing web surfaces, including the main Facebook site. The Web platform carries significantly different constraints from native apps, meaning that we needed to take custom approaches to many different aspects of the product. Further, Web users often add and remove devices in very different patterns from mobile-only users, increasing the complexity of our multi-device challenge.

Learn more about E2EE on Messenger

Today, we are sharing two white papers:

- Our Messenger end-to-end encryption whitepaper, which describes the core cryptographic protocol for transmitting messages between clients.
- The Labyrinth encrypted storage protocol whitepaper, describing our protocol for end-to-end encrypting stored messages history between devices on a user's account.

These add to a number of publications that we have shared which cover Messenger's E2EE, including:

- Our recently updated Safety whitepaper
- The independent E2EE Human Rights Impact Assessment
- Our Security Principles whitepaper
- The Code Verify browser extension

Beyond E2EE for Messenger

The journey to bring E2EE to Messenger has been a long one, but it's not yet finished. While we are globally launching default E2EE for personal one-to-one messages on Messenger, we are still in the testing phase for group messaging and some other products, like Instagram Direct Messages. On Instagram, we are currently testing "disappearing messages" for one-to-one Instagram Direct conversations in select countries. Disappearing messages are ephemeral and, as with those in Messenger, expire 24 hours after being sent. They are built leveraging our E2EE infrastructure and provide an increased level of privacy. We plan to expand this work as well as conduct additional testing around E2EE on Instagram over the next year.