

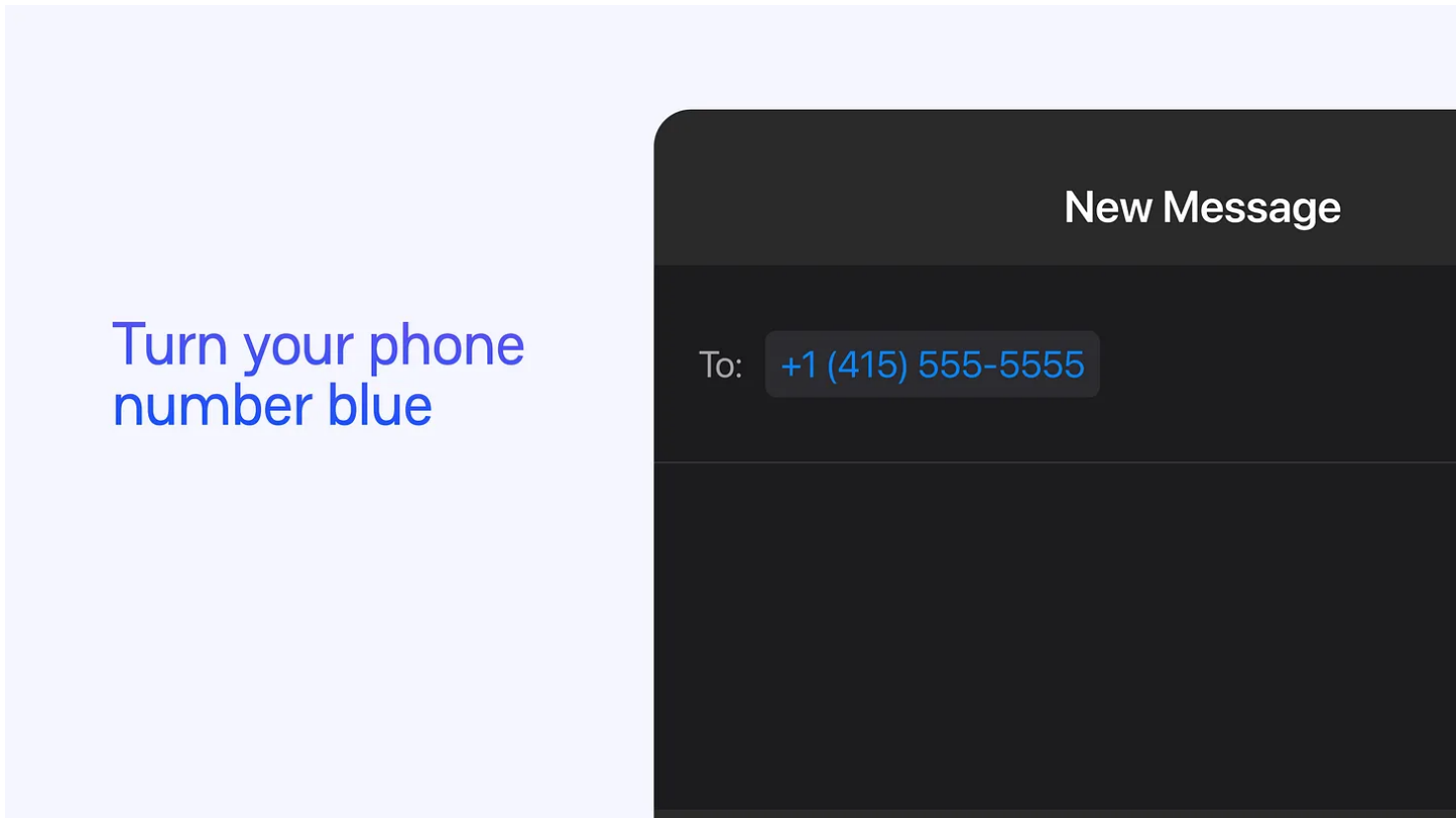
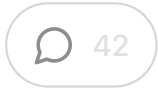
How Beeper Mini Works

It's a technical deep dive, so buckle up!



BEEPER

DEC 05, 2023



We've written this blog post to help you understand how [Beeper Mini](#) works. At Beeper, we believe that it is critical for you to be able to trust the software that you use, especially something as important and sensitive as your chat app. We work to earn and keep your trust in three ways:

1. Transparency - since we started Beeper 3 years ago, we've been taking opportunities like this to explain how Beeper works. We have a proud history of building products, like Pebble, and [stand publicly](#) behind our work.
2. Open source - each major piece of software that we've built to interact with other chat

networks is open source at github.com/beeper.

3. Privacy and security-aligned business model - we make great software and charge a small subscription fee. Simple as that. No ads. Your data stays private.

Security and privacy

Read the entire post for the full story. TLDR: the following features of Beeper Mini ensure that all communication is encrypted and secure.

- All messages are end-to-end encrypted before being sent. Beeper (and Apple) cannot see your messages.
- Encryption keys never leave your device.
- Beeper Mini connects directly to Apple servers. There is no Mac server relay, like other apps.
- No Apple ID is required. Beeper does not have access to your Apple account.
- Your contact list never leaves your device.

Don't believe this is possible? Try the [open-source Python proof of concept](#) on your own computer to see for yourself. Security researchers are invited to verify all claims that we make, see appendix below.

How it works

Beeper Mini works differently than Beeper Cloud in important ways that increase your privacy and security. Beeper Mini is a standalone Android app. It does not require a cloud server to send and receive messages. It also implements [Apple's end-to-end encryption protocol](#) natively within the Android app itself. All messages are end-to-end encrypted before they are transmitted directly from your device to Apple servers. Learn more about iMessage encryption on [Apple Platform Security](#) page.

This is now possible because the iMessage protocol and encryption have been reverse engineered by jjtech, a security researcher. Leveraging this research, Beeper Mini implements the iMessage protocol locally within the app. All messages are sent and

received by Beeper Mini Android app directly to Apple's servers. The encryption keys needed to encrypt these messages never leave your phone. Neither Beeper, Apple, nor anyone except the intended recipients can read your messages or attachments. Beeper does not have access to your Apple credentials.

We built Beeper Mini by analyzing the traffic sent between the native iMessage app and Apple's servers, and rebuilding our own app that sends the same requests and understands the same responses. Learn more by reading jjtech's blog post, [iMessage Explained](#), and his proof-of-concept [Python implementation on Github](#). Anyone can download this code, run it on any computer that supports Python, login to their iMessage account, and send and receive iMessage protocol messages. No Apple hardware required.

Another change is that Beeper Mini does not use the [Matrix](#) protocol, encryption or code like Beeper Cloud. It is a completely new codebase, versus our first Android app, which was a fork of [Element](#). In the future, we are planning to add Matrix network support back in, along with support for the 15 other chat networks in Beeper Cloud. Read more about [our roadmap](#).

Inside the Beeper Mini Android app

1. Sign in

When you first start the Beeper Mini app and sign in with Google, a registration request is sent to our Beeper API Server. This service only exists to verify your subscription status, as well as give our support team the information they need to debug any issues that you may be running into (including your name and email address). No iMessage credentials or messages are transmitted through these servers, which are for Beeper Mini account management only.

2. Permissions and registration

After that, you are prompted to allow notifications, which sends a push token to Beeper Push Notification service, which enables our servers to send push notifications to your Android device. These push notifications do not contain the contents of messages.

Next, you are prompted to grant contact list and SMS permission access.

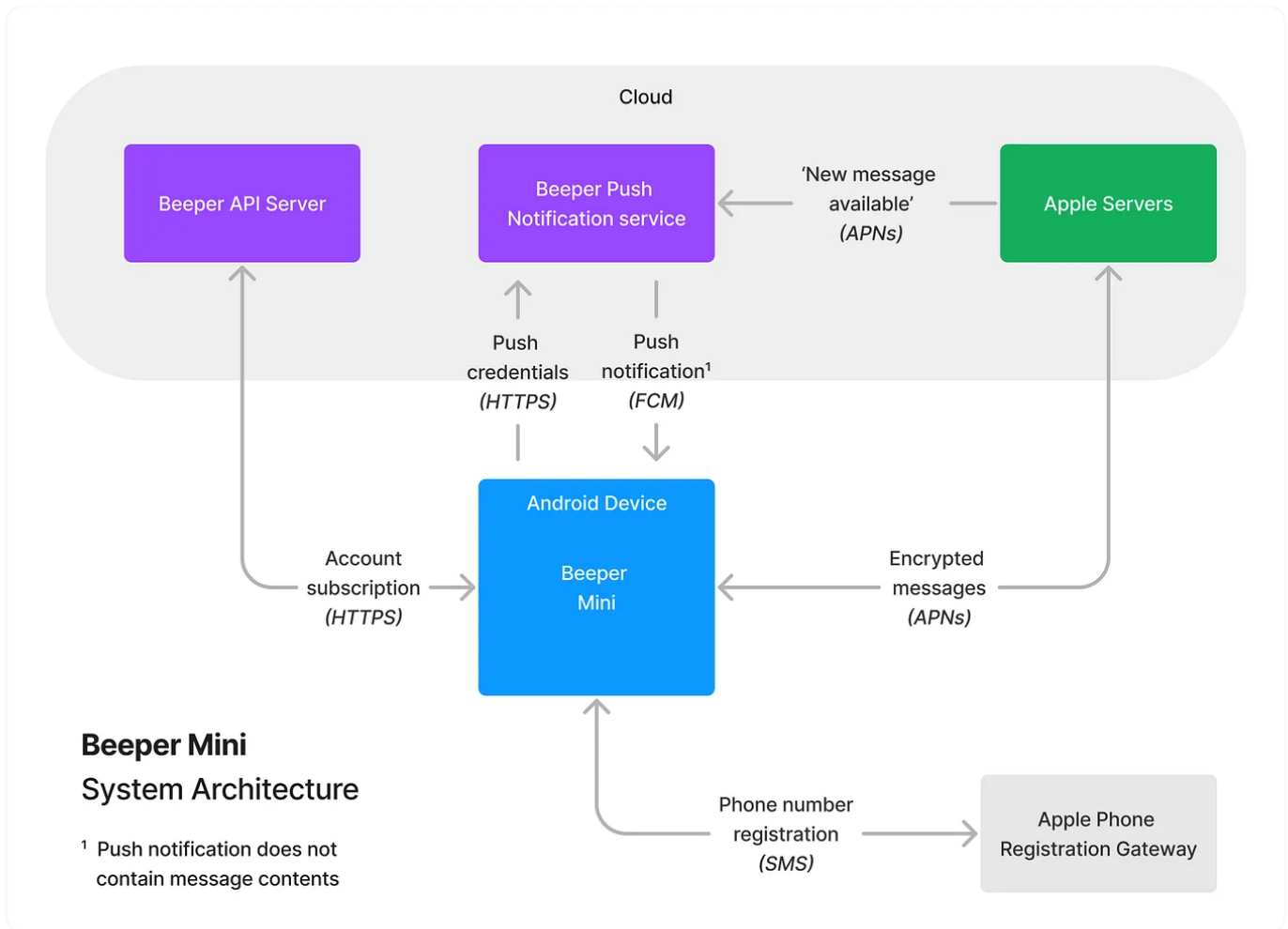
- Contact list access is used to match phone numbers to contact names, and display profile pictures. Your contact list is never sent to Beeper servers.
- SMS access is used to send an SMS text message from your number to Apple's "Gateway" service. The gateway sends a response via SMS, and the contents from that SMS response are sent to Apple to register your phone number as a blue bubble. Your SMS chat history is also used to determine if any of your recent SMS chats were with people who have iPhones. If so, these chats are shown in the inbox.

It's at this point that the app generates encryption keys that are used for end-to-end encrypted messaging. The public key is sent to Apple servers, and the private keys are stored in the Android device local filesystem. Beeper Mini is now signed in.

3. Optional Apple ID sign in

Optionally, you may also sign in to your Apple ID to enable sending/receiving from your email address. This will also enable you to send and receive messages from other Apple devices like iPad or Macs. The Apple ID login sends your username, password and a 2-factor code using encrypted HTTPS requests directly to Apple servers.

4. Sending and receiving messages



Apple's iMessage protocol works over [Apple Push Notification service](#), which most developers would be familiar as the service that allows them to send push notifications to their iOS applications. For iMessage protocol, all messaging traffic flows over this service in both directions, encrypted with keys generated locally on each device. Beeper Mini connects to APNs over TCP, using the credentials generated during the login process.

A persistent connection to APNs is needed to be notified of new incoming messages in real-time. On an iPhone, an APNs connection is maintained by the operating system, and connected at all times. In Beeper Mini, the connection can only be maintained when the app is running, since Android does not support APNs natively.

To work around this limitation, we built Beeper Push Notification service (BPNs). BPNs connects to Apple's servers on your behalf when Beeper Mini Android app isn't running. We can do this while preserving user privacy thanks to Apple separating the credentials

needed to connect to APNs to send and receive content (the “push” credentials) and the keys needed to encrypt and decrypt messages (the “identity” keys). Push credentials can be shared securely with the Beeper Push Notification service, and BPNs can connect to APNs on your behalf. Whenever BPNs receives an encrypted message that it won't be able to decrypt, it simply disconnects from APNs and sends an FCM push notification to wake up the Android app, which then connects to APNs, downloads, decrypts and processes the incoming message. BPNs can only tell when a new message is waiting for you - it does not have credentials to see or do anything else.

BPNs will be notified when you receive a message, but without the encryption keys it can't decrypt anything BPNs receives. Also, without the identity credentials, BPNs can't send messages on your behalf. If you don't mind not receiving real-time push notifications for new messages, your BPNs can be disabled entirely by going to Settings → Manage Connection → Enable Push.

When you create a new chat, the phone number or email address of your intended recipient is transmitted to Apple servers. If the contact is on iMessage, a public key is returned.

Sending messages is even simpler. When you hit send, the message is encrypted with the public keys of the intended recipients and sent directly to Apple servers via an SSL encrypted TCP connection over APNs.

5. Analytics and other services

Beeper Mini connects to a few other services as part of its operation. We use a self-hosted installation of Rudderstack (<https://rudderstack.beeper-tools.com>) for analytics and diagnostic events, which we use for improving the app but can be disabled in Settings → Preferences → Share Diagnostics. We use OneSignal to send education and account related push notifications, and RevenueCat to help integrate Google Play subscriptions.

Other than that, that's it! No other servers or services are used. Beeper Mini keeps your messaging secure by keeping all messaging credentials, keys, messages and media local to your phone, and only sends them directly to Apple's servers after encrypting them with iMessage's end-to-end encryption algorithm.

We value, actually, we treasure feedback. If you run into a bug or have a feature request, there's a button in-app to report a problem. We read every single report.







Brad Murray and Eric Migicovsky

Beeper cofounders







Appendix

To write this blog post, we performed a red team analysis on our own app. We made extensive use of the excellent [mitmproxy](#) project to capture the network traffic coming from a real phone running a modified version of the Beeper Mini client. A modified version was needed for this analysis in order to disable certificate pinning, so that the Beeper Mini Android app would accept being connected to mitmproxy instead of only accepting Apple's certificates for that connection. If researchers would like a copy of this version of Beeper Mini (with cert pinning disabled) to perform a similar analysis, please contact us at security@beeper.com.

Below is a capture of the requests that we make with Apple's servers over HTTPS when logging into iMessage with your phone number. We first register with a service named `albert.apple.com`, which sets up our "push" credentials and allows us to connect to APNS. We then make two requests to get the number we need to send an SMS to register our phone number which is different for each carrier (This capture was taken with a device registered with Rogers, a Canadian cell phone carrier 🇨🇦). Finally, we take the contents of the response SMS (not shown here) and send it to `identity.ess.apple.com`, registering our account with iMessage and generating the "identity" credentials we'll use to send and receive.

Path	Method	Status	Size	Time
 https://albert.apple.com/WebObjects/ALUnbrick.woa/wa/deviceActivation?device=Windows	POST	200	13.8kb	109ms
 https://itunes.apple.com/WebObjects/MZStore.woa/wa/com.apple.jingle.appserver.client.MZITunesClientCheck/version?languageCode=en	GET	200	5.1mb	5s
 https://updates.cdn-apple.com/20230602/carrierbundles/032-23859/3D89D248-1960-4935-8780-07F0BF259703/Rogers_ca_iPhone.ipcc	GET	200	111.6kb	123ms
 https://identity.ess.apple.com/WebObjects/TDIdentityService.woa/wa/authenticatePhoneNumber	POST	200	3.3kb	121ms
 https://profile.ess.apple.com/WebObjects/VCProfileService.woa/wa/idsGetHandles	GET	200	891b	164ms
 https://identity.ess.apple.com/WebObjects/TDIdentityService.woa/wa/register	POST	200	8.4kb	368ms

Optionally, you can also register your Apple ID with Beeper Mini as well, as shown in this capture. You first provide your username and password over encrypted HTTPS directly to Apple's servers, followed by a second request to provide your 2FA code. We can then register for iMessage again, this time providing the certificates from both the earlier phone number registration and our new Apple ID registration. Registering these together in the same call links them together, allowing any other device that you're logged in with your Apple ID to send and receive with both your Apple ID emails and your phone number.

	https://profile.ess.apple.com/WebObjects/VCProfileService.woa/wa/authenticateUser	POST	200	1.1kb	219ms
	https://profile.ess.apple.com/WebObjects/VCProfileService.woa/wa/authenticateUser	POST	200	1.8kb	203ms
	https://profile.ess.apple.com/WebObjects/VCProfileService.woa/wa/authenticateDS	POST	200	5.6kb	214ms
	https://profile.ess.apple.com/WebObjects/VCProfileService.woa/wa/idsGetHandles	GET	200	891b	92ms
	https://profile.ess.apple.com/WebObjects/VCProfileService.woa/wa/idsGetHandles	GET	200	1.4kb	162ms
	https://identity.ess.apple.com/WebObjects/TDIdentityService.woa/wa/register	POST	200	14.1kb	251ms

Next, a capture of the keys shared with the Beeper Push Notification service (hostname `imux.beeper.com`). Note, the RSA private key in this request is your “push” credentials that allow you to connect to APNs, not your “identity” credentials that allow you to encrypt and decrypt iMessages. Push credentials cannot be used to escalate permissions or access anything other than the presence of a new APNs push notification. Check out [apns.py](#) in [pypush PoC](#) to learn more about push credentials.

PUT https://imux.beeper.com/v1/apns/device HTTP/2.0

user-agent: imessagego

authorization: Bearer

imat_51efd3n9enyukwr97khfrtnrddrkvmkm16n7n1ufm9ddw6y7e140

content-length: 3254

accept-encoding: identity

JSON

Edit

Replace

View: json ▾

```
{
  "apns": {
    "cert": "-----BEGIN CERTIFICATE-----\nMIIDdzCCAuCgAwIBAgIKAYdc
    "key": "-----BEGIN RSA PRIVATE KEY-----\nMIIEpAIBAAKCAQEA/ReLX
    "token": "YQ2kV8xIAACxiLRHZhtrxQXLl8d9jTzRsy/n+ZSD8hg="
  },
  "fcm": {
    "token": "dtis9GENRzqd9dTY2mQMMS:APA91bFrS CrRM6qIH2st4sAWMEAHc
  }
}
```

Sending and receiving is not shown here, as they are not done over HTTP but instead through an SSL encrypted TCP connection to APNs. The APNs servers are hosted at `*-courier.push.apple.com`, where the asterisk is replaced by a number between 1 and 30. All message contents and media are encrypted with your “identity” keys, which never leave your Android phone.

There is a `/login` endpoint on Beeper servers, but as mentioned previous, this is only for subscription management purposes. The client submits the token received from the Google login process to our servers, and the response contains their subscription status. No iMessage credentials are ever sent to Beeper servers.

POST https://api.beeper.com/ima/login HTTP/2.0

user-agent: Beeper/1.0.0 (Google Pixel 3 XL; Android 12; SP1A.210812.016.C2)

accept: application/json

accept-charset: UTF-8

content-type: application/json

content-length: 1129

accept-encoding: identity

JSON

Edit

Replace

View: auto

```
{
  "token": "eyJhbGciOiJIJSUzI1NiIsImtpZCI6ImU0YWVmYjQzNmI5ZTE5N2UyZTEwMDZhZjJjODQyMjg0ZTQ5ODZhZmYiLCJ0eXAiOiJK"
}
```

JSON

Edit

Replace

View: auto

```
{
  "access_token": "imat_51efd3n9enyukwr97khfrtnrddrkvmkm16n7n1ufm9ddw6y7e140",
  "analytics_id": "cl_xnz67krf70",
  "ima_user_token": "imau_f878geu2ar",
  "subscription": {
    "active": true,
    "expires_at": "2223-10-08T00:39:43Z",
    "product_identifier": "rc_promo_imessage-on-android_lifetime"
  }
}
```

Note: Beeper and Beeper Mini are entirely independent software products, with no relationship to, or endorsement by, Apple, Google, or any other supported chat networks.

iMessage, Apple, Mac and iPhone are trademarks of Apple, Inc.

Android is a trademark of Google, LLC.



19 Likes · 4 Restacks