



## Technology preview: Sealed sender for Signal

jlund on 29 Oct 2018

In addition to the end-to-end encryption that protects every Signal message, the Signal service is designed to [minimize the data that is retained about Signal users](#). By design, it does not store a record of your contacts, social graph, conversation list, location, user avatar, user profile name, group memberships, group titles, or group avatars.

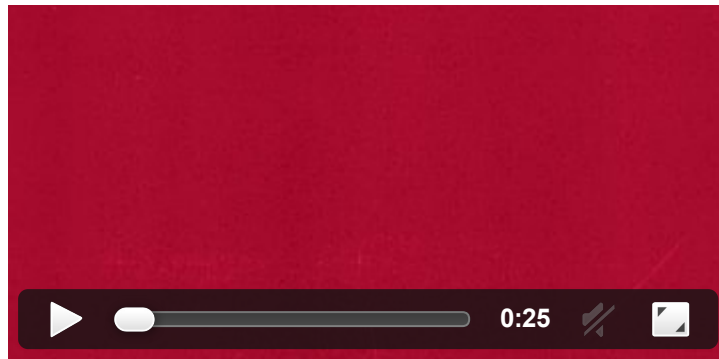
We have been [exploring techniques](#) to further reduce the amount of information that is accessible to the service, and the latest beta release includes changes designed to move Signal incrementally closer to the goal of hiding another piece of metadata: who is messaging whom.

### Pushing the envelope

When you send a traditional piece of physical mail, the outside of the package typically includes the address of both the sender and the recipient. The same basic components are present in a Signal message. The service can't "see into" the encrypted package contents, but it uses the information written on the outside of the package to facilitate asynchronous message delivery between users.

While the service always needs to know where a message should be delivered, ideally it shouldn't need to know who the sender is. It would be better if the service could handle packages where only the destination is written on the outside, with a blank space where the "from" address used to be.





Traditionally, a Signal client sends a message by connecting to the service over TLS, authenticating, and handing off the encrypted message contents along with a destination. The authentication process serves two primary functions:

1. The service can validate the sender's identity to help prevent spoofing and provide the recipient with some assurance about who sent the message.
2. The service can use the sender's identity to apply rate limiting and abuse protection.

In order to remove the “from” address from the outside of the package, we need a new process that will account for these two existing functions.

## Special delivery

### Sender certificates

To prevent spoofing, clients periodically retrieve a short-lived sender certificate from the service attesting to their identity. The certificate contains the client's phone number, public identity key, and an expiration timestamp. Clients can include the sender certificate when a message is sent, and receiving clients can easily check its validity.

### Delivery tokens

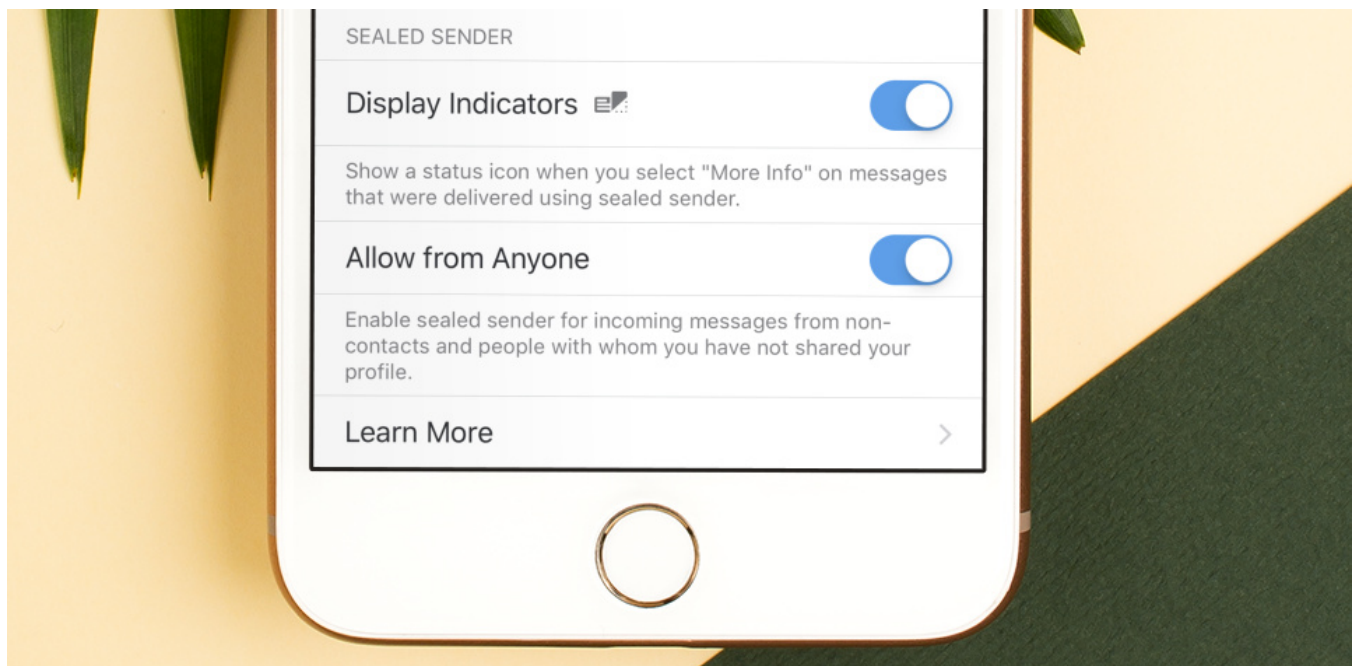
To prevent abuse, clients derive a 96-bit delivery token from their profile key and register it with the service. The service requires clients to prove knowledge of the delivery token for a user in order to transmit “sealed sender” messages to that user.

As a reminder, [Signal Profiles](#) are end-to-end encrypted, and the profile keys are exchanged via the same Signal Protocol messaging channel that protects conversations and calls. Profiles are shared with your contacts, other people or groups who you explicitly approve, and in conversations that you create. Building on this existing foundation allows delivery tokens to be seamlessly exchanged behind the scenes.

Since knowledge of a user's profile key is necessary in order to derive that user's delivery token, this restricts “sealed sender” messages to contacts who are less likely to

require rate limits and other abuse protection. Additionally, blocking a user who has access to a profile key will trigger a profile key rotation.

On the opposite end of the spectrum, users who want to live on the edge can enable an optional setting that allows them to receive incoming “sealed sender” messages from non-contacts and people with whom they *haven't* shared their profile or delivery token. This comes at the increased risk of abuse, but allows for every incoming message to be sent with “sealed sender,” without requiring any normal message traffic to first discover a profile key.



## Encryption

Message contents are end-to-end encrypted with the Signal Protocol as normal, but the “envelope” containing the sender certificate as well as the message ciphertext is then *also* encrypted using the sender and recipient identity keys:

```
message_ciphertext      = signal_protocol_encrypt(message)
e_pub, e_priv           = X25519.generateEphemeral()
e_chain, e_cipherKey, e_macKey = HKDF(salt="UnidentifiedDelivery" || recipientIdentityPublic)
e_ciphertext            = AES_CTR(key=e_cipherKey, input=senderIdentityPublic)
e_mac                   = Hmac256(key=e_macKey, input=e_ciphertext)

s_cipherKey, s_macKey = HKDF(salt=e_chain || e_ciphertext || e_mac, ikm=ECDH(recipientIdentityPublic, e_priv))
s_ciphertext           = AES_CTR(key=s_cipherKey, input=sender_certificate || message_ciphertext)
s_mac                  = Hmac256(key=s_macKey, input=s_ciphertext)

message_to_send = s_ciphertext || s_mac
```

## Sending a message

Putting it all together, the process of sending a “sealed sender” message looks like this:

1. Encrypt the message using Signal Protocol as usual.
2. Include a sender certificate in the envelope.
3. Encrypt the envelope to the recipient.
4. Without authenticating, hand the encrypted envelope to the service along with the recipient’s delivery token.

The recipient of the message can then decrypt the envelope, validate that the identity key which was used to encrypt the envelope matches the sender certificate, and continue processing as normal.

## The future is in transit

As clients upgrade, messages will automatically be delivered using sealed sender whenever possible. Users can enable an optional status icon that will be displayed in the detailed information view for a message to indicate when this happens.

These protocol changes are an incremental step, and we are continuing to work on improvements to Signal’s metadata resistance. In particular, additional resistance to traffic correlation via timing attacks and IP addresses are areas of ongoing development.

We [do not collect or store](#) any sensitive information about our users, and that won’t ever change.

## Deliver us from beta

Our beta releases are not for the faint of heart. If you need a stable and reliable messaging experience, we strongly encourage you to wait. However, if you’re ready for a life of adventure (and occasional sadness), we invite you to [join the Signal public beta](#) and help us test the latest release on your favorite platform.

Beta releases that support sealed sender will be rolling out over the next few days. Be sure to update all of your devices (including linked Desktops). We’re excited to hear what you think!

**Want to get involved with Signal? [We're hiring!](#)**

## **Company**

[Donate](#)

[Careers](#)

[Blog](#)

[Terms & Privacy Policy](#)

## **Download**

[Android](#)

[iPhone & iPad](#)

[Windows](#)

[Mac](#)

[Linux](#)

## **Social**

[GitHub](#)

[Twitter](#)

[Instagram](#)

## **Help**

[Support](#)

[Community](#)

© 2013–2021 Signal, a 501c3 nonprofit.

Signal is a registered trademark in the United States and other countries.

For media inquiries, contact [press@signal.org](mailto:press@signal.org)