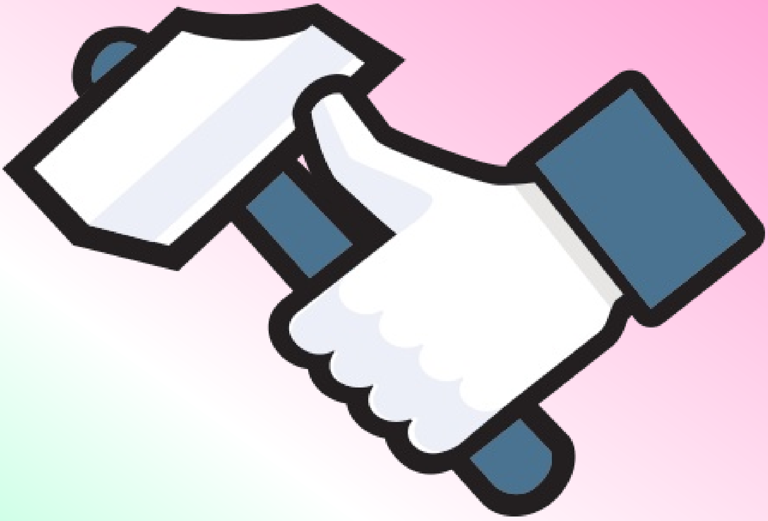
SPECIAL ISSUE



IMMATERIAL LABOUR UNION

XMPP

#9

Editors' Note

homebrewserver.club

Welcome to the very first special issue of the Immaterial Labour Union zine! After the announcement from WhatsApp in August 2016 regarding the sharing of private user information with Facebook, the homebrewserver.club (HBSC)—a Rotterdam based initiative hell-bent on seizing the means of serving—convened with the purpose of collectively exploring and setting up decentralised instant messaging alternatives. Following a philosophy that puts a higher premium on approaches, instead of apps, the HBSC decided to explore the possibilities for federation offered by XMPP, an instant messaging communication protocol designed as an open standard. Those meetings resulted in the desire to write an extended article on the world of private messaging services, as well as the problems of scale and trust, and so this special issue was born. Besides continuing the Immaterial Labour Union's longstanding tradition of looking at the issue of privacy from a labour perspective, this issue offers you hands-on guides on how to set up your own XMPP server and XMPP Messenger (both desktop & mobile). Last but not least, at the end of this special issue, you can find an XMPP glossary that we put together to give a better understanding of all the jargon involved. You can also read these articles on the homebrewserver.club official website[1]!

About the Homebrew Server Club
The Homebrew Server Club is a monthly gathering for those who (wish to) host their own online services from home, rather than using commercial and privacy unfriendly alternatives.

1: https://homebrewserver.club/

2: https://creativecommons.org/licenses/by-sa/2.0/

Have you considered the alternative

homebrewserver.club

> "Remember, when advertising is involved you the user are the product. (...) When people ask us why we charge for WhatsApp, we say 'Have you considered the alternative?'"
>
> *Brian Acton and Jan Koum, June 2012*[1]

> "Facebook today announced that it has reached a definitive agreement to acquire WhatsApp, a rapidly growing cross-platform mobile messaging company, for a total of approximately $16 billion, including $4 billion in cash and approximately $12 billion worth of Facebook shares."
>
> *Facebook Newsroom, February 2014*[2]

> "By coordinating more with Facebook, we'll be able to do things like track basic metrics about how often people use our services and better fight spam on WhatsApp. And by connecting your phone number with Facebook's systems, Facebook can offer better friend suggestions and show you more relevant ads if you have an account with them." *Brian Acton and Jan Koum,*
>
> *August 2016*[3]

Pattern Recognition

WhatsApp started out full of dreams: "we want WhatsApp to be the product that keeps you awake...and that you reach for in the morning. No one jumps up from a nap and runs to see an advertisement"[4]. When they thought of WhatsApp, Brian Acton and Jan Koum were very keen on not selling our user data for targeted advertisement purposes. So they charged a nominal rate for the use of their service, rightfully pointing out the hidden cost of using free services.

In the year of 2014 however, WhatsApp was bought by Facebook, thus joining the social network's happy and expanding family of venture capital investments, a family including Instagram, purchased in

April 2012, and Oculus VR, purchased the month before. At the time, many, and with good reason, worried about the changes this acquisition could entail for WhatsApp. Eventually, in August 2016, WhatsApp users everywhere learned about what was in fact unavoidable. The company that built its reputation upon an ad-free ethic, would now be sharing private user information with Facebook, its parent company. So we, the users, are the product after all, and as expected, this is presented in the form of an *improvement* of the user experience. Thanks to the tighter coordination between WhatsApp and Facebook, we can now more easily find our friends or see more valuable messages from the companies that truly matter to us. Of course, small footnote, these 'benefits' comes at the price of sharing our phone number and other private data with Facebook—though, trusting their word, not the content of the messages themselves.

Facebook does this for the simple reason that it needs to increase its market share on mobile devices[5]; the family of Whatsapp, Facebook and Instagram are all *different* channels leading to this same purpose. One of the consequences of this is that while Facebook's chat function can still be used on their mobile website, plans are that we will soon be forced to install Facebook Messenger should we wish to continue using it on our mobile phones[6]. Once again, in a stroke of pure genius and creativity, this move is being marketed as a way to provide us with the best experience ever. And we can use it with just a phone number, we don't even need a Facebook account. That way, their user base expands along with their profits.

Every time there is a breach of user trust —read: a change in the Terms of Service— or news regarding network surveillance, people are on the lookout for an alternative, and rightfully so. In these moments there are many also willing to promote such alternatives, usually in the form of yet another disruptive app. After the purchase of Whatsapp, for example, Telegram was advertised as the alternative. After it became clear that Telegram had dreadful security, people promoted Viber. Then Snapchat, then Threema, then Allo and now Signal. There is a reason why we're falling into this pattern of needing alternatives to the alternatives. And that is because...

There are no alternatives.

There’s a tendency to oversimplify the issues related to the use of these apps as merely a privacy matter, and not even that is sufficiently addressed. While each of the aforementioned apps are alternative companies and brands, what these alternatives all have in common is that they share the same model. A model that revolves around centralized services, vendor lock-in and marketing related surveillance, and all of that within a neoliberal context of the free market. These alternatives therefore promote themselves as more than just an alternative, but also as competing products, usually highlighting a particular feature lacking in rivals’ products. Remember that ill-fated, super cool, nice looking alternative to Facebook, Ello? It gained a lot of traction out of legitimate concerns with Facebook’s modus operandi, promoting itself as an alternative for its nice features and its promise not to use advertising. But as Aral Balkan was quick to point out[7], allowing investments by venture capital firms meant the project was dead before it really began. Taking these investments, which allowed them to scale as a platform, also meant that they would, at some point, have to make a lot of money for their investors. How? By selling ad space or information about their users. The reason the pattern keeps repeating itself is not because the makers of these apps always secretly intended to sell your data while saying they wouldn’t. The reason is that they have no choice within the economic system they choose to operate in.

Cryptography matters, but then it also doesn’t

The latest competitive feature—one might even say, marketing trick—to make concerned users switch from one alternative to another is cryptography, the act of coding messages during communication. This strategy works well because the vast majority of people are not really informed when it comes down to the technicalities of cryptography, so this discourse mostly serves to throw bedazzling sparkles in our eyes. To be sure, cryptography is fundamental for privacy. However, the main privacy threat in the context of using these apps isn’t the potential of a government eavesdropping on our communications. The privacy threat is the wholesale and increasing dependence on centralized services which revolve around the surveillance and monetization of user information. In 2016, both WhatsApp and Facebook Messenger enabled end-to-end encryption to address increasing privacy concerns. Adding *crypto* to a communication app in this case merely obfuscates a concern about the hegemony of these platforms. In essence, the issue of privacy is much larger than just the lack of cryptography; the conditions that threaten privacy are structural and economic and not resolved by a *patch* or a new feature.

This issue is further stressed when looking at the question of metadata, that is to say, data about data, which in the case of communication applications is everything but the communication data itself. When WhatsApp started sharing, among other things, its users' phone numbers with its parent company, Facebook, it went to great lengths to guarantee us that the content of our messages was still perfectly secure, impossible to be read by both WhatsApp and Facebook. The argument stating that "It's only metadata, don't worry" has been however debunked numerous times. Even though these platforms would love us to believe otherwise, metadata is neither a trivial disposable by-product, nor it is anonymous. And assuming that the crypto is sound and that the app running this crypto is not flawed, cross-referencing several databases containing metadata will always produce an array of very personal information, that in itself is much more valuable than encrypted naked selfies. Thus it should be no surprise that former NSA director Michael Hayden infamously said in 2012 "we kill based on metadata"[8] and later argued in 2015 that metadata should be the main area of focus of surveillance activities, and not the creation of backdoors within crypto, or the banning of the latter[9].

In short, both Whatsapp and Facebook Messenger can afford to deploy end-to-end encryption for your messages because it won't hurt their bottom line, which is making money based on the surveillance of your behavior and your social graph. Adding crypto thus merely patches your privacy worries, but not the threat to it.

The Wrong Signal[10]

The end-to-end encryption enabled in WhatsApp and Facebook Messenger has been developed by Open Whisper Systems, a non-profit run by crypto-celebrity Moxie Marlinspike. OWS also developed the algorithm for their own instant messaging application, Signal, and then open-sourced it. Signal itself is now the latest app being promoted as an alternative to WhatsApp and is hailed as the panacea of both security and usability. It even has the backing of members of the dissident elite such as Edward Snowden.

While OWS provides thorough expertise in the field of cryptography, Marlinspike is currently advocating centralisation as the only answer towards user-friendly, fast and secure messaging apps. Decentralisation, according to him, has no place in the modern world and apparently hampers innovation. However, some of his arguments have not remained unchallenged. In particular, where Marlinspike accuses federation of stalling evolution[11], Daniel Gultsch provides a counter argument by using the Web as an example of successfully federated system[12]. Furthermore, Gultsch states that the problem is not that federation doesn’t adapt, but rather that there are problems with its implementation for a very significant reason: software developers working on federated systems mostly work for free in their spare time or with little means, given the difficulty to monetise a system which design can only succeed if it is open and can be appropriated easily beyond its original scope, and thus making its capitalisation particularly challenging. In that sense, the most interesting aspect of this debate is that while Marlinspike seems to defend his product from a technological perspective, Gultsch’s counter argument moves back the discussion to the context of political economy.

Daniel Gultsch is an important counter-voice because he is the main developer behind Conversations. This open-source instant messaging app tries to be both accessible for new users as well as provide enough flexibility for more advanced users. In that regard, Conversations itself does not manage to escape the logic of competition and the discourse around alternative superior apps discussed previously. However, its approach is significantly different because unlike any other apps, Conversations is not a complete solution, nor does it present itself as such. It is a client that relies on federation, which means that it allows for people to chat with each other regardless of what provider they are using. In concrete terms, there is no central server directly connected to Conversations, but Conversations can connect to different chat servers. This is possible because Conversations is built upon a long-lived messaging protocol called XMPP.

XMPP, the federated messaging protocol

Up to a few years ago XMPP and its implementations were lagging behind in terms of mobile features, usability and interface design[13]. That was the so-called lack of evolution Moxie pointed out. But recently Gultsch and the other contributors to Conversations have managed to bring XMPP up to speed with the functionality of well known mobile messenger applications. Not only did this demonstrate that bridging the gap could be done technically, but it also had the effect of breathing new life into the XMPP community. An example of this new energy was the initiative to create and implement OMEMO, an XMPP Extension Protocol that provides multi-user end-to-end encryption and which is based on Signal's own encryption algorithm. Ever since a growing number of clients have started implementing OMEMO[14], including Gajim for desktops and ChatSecure for iPhones.

Gultsch succeeded: his XMPP client Conversations has been installed between 10 and 50 thousand times[15] and he is able to live off and work full-time on the project so far precisely because of understanding the technical underpinnings of centralized services such as WhatsApp or Signal. It is however a bitter-sweet victory, because as Gultsch articulated in his defense of decentralisation, the main difference between centralised and decentralised implementations is not only technical, but also a matter of economic sustainability. In other words, if his ongoing efforts show that it is possible to have a satisfying and safe user experience while using federated alternatives, this is only possible because, unlike any other XMPP client developers, he is in the position of working on this project full time. The problem has not been solved but shifted. If economically sustainable XMPP federation were to scale to the point of being as successful as the centralised solution offered by Signal, it would have to face the consequences of doing so in the context of a free market driven by competition. In that situation, each XMMP client's economic viability would depend heavily on its capacity to capture enough users that can provide income for their developers. The problem therefore is not so much a problem of the technical or economical sustainability of federation, but more a problem of the economic sustainability of open standards and protocols in a world saturated with solutionist business models. After all, many years ago, Google and Facebook did provide XMPP support in their chat applications before deciding to close its interoperability.

Approaches not Apps

Given the different problems mapped in this text, it becomes difficult to blindly recommend Conversations as the superior alternative, that is to say, a near drop-in replacement to Signal or any other competing secure communication software.
The reason is not technical but is linked to the fact that, as discussed earlier, Conversations' own success relies on an economic model that is quite fragile, and the success of which—and it's a paradox—could potentially undermine the cultural diversity of the XMPP ecosystem. With that said, there are however two essential points that the Conversations case brings up. These points are not always articulated clearly in discussions on federation: scale and trust.

Rather than having to swap one app for the other in an attempt to mitigate a large and confusing privacy problem, the XMPP federation approach allows to collectively tackle the problem based on its various discrete parts. Such an approach, rather than suggesting a singular and proprietary solution, allows for the existence of different free and open source software servers which can be combined with different free and open source software clients. That makes it possible for you and a group of friends to run your own infrastructure, whether on a rented server or on a very small home server. The federated nature of the protocol allows you to try, play and experiment with different network infrastructures with different clients. These clients can range from custom XMMP bots to general instant messengers that you would be able recommend your friends and family to replace Whatsapp, without making a fool of yourself. As these open-source technologies continue to evolve you can make incremental changes to your server or switch clients as newer versions arrive. Hosting your own infrastructure allows you to scale your communication in a way that is the most meaningful for the group or community you belong to. It is also a way to make sure your system matches your own threat model, while simultaneously allowing you to deal with trust that is not mediated by an app. It also allows you to experiment with economic models other than those linked to large-scale infrastructure involving surveillance and capturing of your social graph for financial gain. Maybe you want to share the cost of the server or the responsibilities of administrating it, maybe you want to collectively learn how to run all this stuff, or maybe you want to start meetings to exchange tips, etc. However, this does not mean that you need to cut yourself off from the rest of the world and this form of localism should not be misunderstood for a hipsterist and reactionary form of escapism. Instead, such an approach is quite the opposite as it provides a possibility to actively engage with societal issues. It allows groups to collectively think, in the sense of defining questions and hypotheses themselves, acquire skills and knowledge and respond to issues that are both relevant to their own situation but that can also resonate globally, enabling others to start a similar process.

The goal of this article was to provide some tools and insights which not only allow for contextualisation of the technology we are using and supporting, but also help making sure that the instant-messaging you and your friends use happens in a trusted and secure environment, as much as possible outside the economies of surveillance. For this reason our motivation for writing this article was two-fold. On the one hand we wanted to show that the issue of privacy is more insidious than institutional eavesdropping and not merely solved with the use of end-to-end encryption. On the other hand, and as a consequence, we wanted to suggest not a different app, but a different approach altogether on the basis of XMPP federation and collective action. Therefore we’ve written two guides. One on how to configure a server and one on how to choose and use clients that can go along with it. These allow you to put a self-hosted approach, an approach that brings aspects of trust, scale and implementation to the forefront and into practice. Once again, such guides should not be perceived as definitive answers but more as tools to keep us, and hopefully you too, busy formulating the right questions and building networks of mutual help. So while we are unable to recommend you the next big app that will solve all user surveillance and financialisation once and for all—as we are pretty sure no such app will ever even exist—we hope to at least help shed a light on the confused and confusing discourses that surround crypto-sound alternatives which may obfuscate less obvious problems.

1: https://blog.whatsapp.com/245/Why-we-dont-sell-ads

2: https://newsroom.fb.com/news/2014/02/facebook-to-acquire-whatsapp/

3: https://blog.whatsapp.com/10000627/Looking-ahead-for-WhatsApp

4: https://blog.whatsapp.com/245/Why-we-dont-sell-ads

5: https://www.theguardian.com/technology/2016/aug/25/whatsapp-to-give-users-phone-number-facebook-for-targeted-ads

6: https://www.theguardian.com/technology/2016/jun/06/facebook-forcing-messenger-app-explainer
7: https://2018.ar.al/notes/ello-goodbye/
8: https://www.youtube.com/watch?v=UdQiz0Vavmc
9: https://www.c-span.org/video/?402284-1/discussion-immigration-policy-national-security
10: https://it-kollektiv.com/wrong-signal-das-falsche-signal-engl/
11: https://signal.org/blog/the-ecosystem-is-moving/
12: https://gultsch.de/objection.html
13: https://op-co.de/blog/posts/mobile_xmpp_in_2014/
14: https://omemo.top/
15: https://play.google.com/store/apps/details?id=eu.siacs.conversations&hl=en

Configuring an XMPP server

homebrewserver.club

This is a guide to set up a modern XMPP server focused on security and mobile messaging. The whole guide further assumes one is using Debian as a server and that you will end up hosting a few of your friends. It further assumes you have some basic skills working on a linux command line.

To make your server communicate make sure following ports are open in your firewall:

```
5222 (for client to server)
5269 (server to server)
5280 (default http port for prosody)
5281 (default https port for prosody)
```

Enabling HTTPS

First we acquire a signed HTTPS-certificate via Let's Encrypt[1]: This is among others required for Gajim plugins to work properly; self-generated certs will not work.

Install Certbot and get new certificates for your domain (replace myserver.org with your own):

```
wget https://dl.eff.org/certbot-auto
chmod a+x certbot-auto
certbot-auto certonly -d muc.placeholderdomain.org -d
dump.placeholderdomain.org -d placeholderdomain.org-d
placeholderdomain.org
```

Should you succeed, you will be able to read something like:

"Congratulations! Your certificate and chain have been saved at /etc/letsencrypt/live/placeholderdomain.org/fullchain.pem. Your cert will expire on 2017 02-13. To obtain a new or tweaked version of this certificate in the future, simply run certbot-auto again. To non-interactively renew *all* of your certificates, run *certbot-auto renew*"

Take note of the path where the certificate is stored as we will use it later.

Installing and setting up MySQL as a storage back-end

First update your repositories and install MySQL:

```
apt-get update && apt-get install mysql-server
```

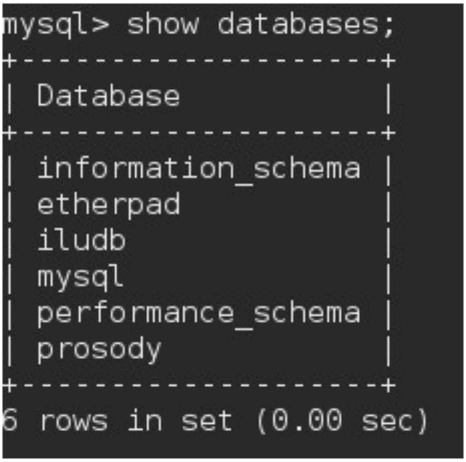
Run mysql as the root user:

```
mysql -u root -p
```

In mysql:

```
mysql> create database prosody; mysql> show databases;
```

Result should be something like:



```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| etherpad           |
| iludb              |
| mysql              |
| performance_schema |
| prosody            |
+--------------------+
6 rows in set (0.00 sec)
```

Create a database account for prosody:

```
mysql> create user prosody;
```

Give the user prosody the rights to access the database, make sure to change the password and take note of it:

```
mysql> grant all on prosody.* to 'prosody'@'localhost' identified by 'userPassword';
```

Exit mysql:

```
exit;
```

Installing and configuring Prosody, the XMPP server

Install the newest version of Prosody and its dependencies from the official prosody repository:

```
echo "deb http://packages.prosody.im/debian wheezy main" >> /etc/
apt/sources.list
wget https://prosody.im/files/prosody-debian-packages.key -O- |
sudo apt-key add -
apt-get update && apt-get install prosody lua-dbi-mysql lua-zlib
```

Add the Let's Encrypt Certificates to Prosody and make sure Prosody can use them:

```
cp /etc/letsencrypt/live/myserver.org/*.pem /etc/prosody/certs/
```

Make sure the certificates are owned by prosody and legible only by root:

```
chown -R prosody:prosody /etc/prosody/ chmod -R 700 /etc/prosody/
certs/
```

Install the newest prosody plugins:

```
apt-get install mercurial cd /usr/src hg clone https://
hg.prosody.im/prosody-modules/ prosody-modules
```

Make a backup of the default prosody configuration and install the one by the homebrewserver.club[2]:

```
cd /etc/prosody
cp prosody.cfg.lua prosody.cfg.lua.original
wget http://homebrewserver.club/downloads/prosody.cfg.lua
```

Replace all instances of the placeholder domain name and passwords in the config file with your own:

```
sed -i 's/placeholderdomain.org/yourdomain.net/g' prosody.cfg.lua
&& sed -i 's/userPassword/yourownpassword/g' prosody.cfg.lua
```

Alternatively you can change them by hand. They are on line 61, 69, 72, 75 of prosody.cfg.lua

Finishing up

After you've set up all of the above it is time to start the server:

```
/etc/init.d/prosody restart
```

Users can be added from the command line, you will also be prompted for a password:

```
prosodyctl adduser me@placeholderdomain.org
```

Alternatively you can change "allow_registration = false;" to "allow_registration = true;" in the config (line 35) to allow users to register accounts on your server via their clients.

Now you can try connecting to your own server by using a client like Gajim or Conversations. Login with the above configured username and password.

If you have questions about Prosody, the project's documentation[3] is quite good. If you can't find answers there, try contacting prosody developers and users directly via the Prosody XMPP chatroom[4].

1: https://certbot.eff.org/
2: homebrewserver.club/downloads/prosody.cfg.lua
3: https://prosody.im/doc
4: xmpp://prosody.conference.prosody.im?join

Setting up an XMPP Messenger

homebrewserver.club

Introduction

First of all, we recommend to follow this guide with a friend who also wants to use this technology. Then you can help each other out and immediately test if everything works. Aside from that, there are already enough XMPP users with empty contact lists around :'(

Parts of this guide on are based on articles written by Mathias Renner[1] on how to set-up Conversations[2] and Gajim[3].

This is a guide for Conversations (Android), ChatSecure(iOS) and Gajim(desktop Win/Lin)

Registering an account

To begin with, you will need a XMPP account! These look like e-mail addresses (username@servername.com) and can be registered with XMPP servers. To get such an account there are a few options:

Host your own XMPP server at home.

If you don't want to or can't set up your own XMPP server:
- Find a friend who runs an XMPP server and ask her for a user account! - Have a look at the list of public XMPP servers[4]. There are some things to look out for however. Make sure they are compliant with all the modern server extensions[5] or you won't get the user experience you are looking for. It is also possible to test other servers yourself using this compliance tester[6]. Also look out for services requiring your phone number to register, it is not needed for XMPP and it would defeat the purpose of taking back the reins of your messaging horse. - Register an account with Conversations.im[7], the server run by the developer of Conversations. It is well maintained and running the latest features: The first six months are free, afterwards there is a monthly fee that goes to support and sustain the ongoing labour into Conversations and free open-source messaging protocols.

Considering XMPP clients

Then you need to pick and install a client (also known as an app or a software package). There are many clients available that support XMPP chats, both for mobile, desktop and web-based environments. The nice thing about using XMPP is that your account and your client are not intertwined, as is the case with Whatsapp, Telegram, Signal and the others. These applications offer a full chat service, which includes the facilitation and hosting of your messages over the network, and the interface options of your client. By separating the two, you have the option to choose. To pick from all the available clients we made a list of criteria of what we considered essential requirements and started crossing off all those applications that didn't meet them:

- Free & open source software — the technology is open, and therefore it's possible to install use the software on your own terms.

- Works with federated servers — servers are not all controlled by a single company or organization, but can also be run by volunteers, organizations, companies, you and me.

- Highly secure (which means support for encryption)] — the software takes security to heart and offers things like end-to-end OMEMO encryption.

- The project is recently updated — There are many XMPP clients available, but not all of them are still maintained. For example: many iOS clients have not been updated for a long time.

- Support for easy image sharing — Essential in order to be able to share dank memes and food pictures.

- Relative ease of use - Need we explain more?

This (apparently) rather rigorous list of requirements left us with three applications that we will discuss in this guide: Conversations for Android, ChatSecure for iOS and gajim for the desktop computer. There are many other XMPP clients however, and while most of these did not meet our requirements for use, they might be ok for you. Have a look at this extensive list of XMPP clients in general[8] and this list of clients[9] that (plan to) support OMEMO. Additionally you might want to make sure your client supports some of the 'modern' XMPP Extensions XEP-0163: Personal Eventing Protocol[10] (for avatars and OMEMO), XEP-0198: Stream Management[11](for better experience using flaky mobile connections), XEP-0280: Message Carbons[12] (sync messages between your different clients), XEP-0313: Message Archive Management[13] (receive messages while offline)], XEP-0363: HTTP File Upload[14] (send images, share files in groupchats and with offline contacts.).

Conversations, Android

- Download the Conversations app on your Smartphone -

Conversations is available via Google Play for €2,39. The sale of the app goes towards the ongoing development of the software.

In case you don't use Google apps or want it for free, you need to install the alternative app store f-droid[15] before. F-droid works like the app store Google Play, except that it isn't a store and only offers apps that are free and open source software. See instructions in the next paragraph how to install f-droid.

If you decided for f-droid, open the website with your phone's browser. Press the big download button on the website, which will download f-droid’s installer. After download, press the downloaded file and the installer should start. Next, start f-droid, update the repositories and search for the app Conversations.

- Start the messenger app and register/log in -

Now, start Conversations. If you already have an XMPP account, you can log in with your so-called JID (jabber id, username@server.com)] and password. Otherwise, if your server of choice has the option for application-based registration enabled, it is also possible to register a new account in this menu, by selecting the "register new account on server" option.

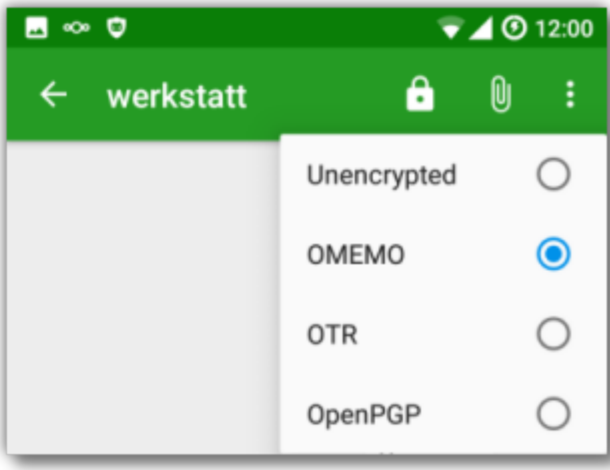
After you clicked Next, the registration process might take up to 20 seconds.

- Start chatting -

To start a chat you need to add another Jabber friend under the '+' in the menu and insert your friend's Jabber ID, e.g. your-friend@a-jabber-server.com. That's it. You can now chat with your friend. However, this will be unencrypted!

- Encryption -

So let's activate OMEMO encryption by pressing the padlock in the top menu bar:



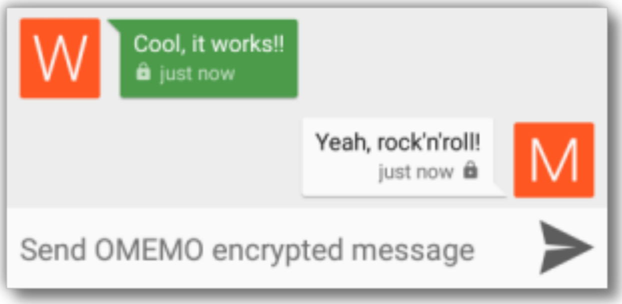
Selecting OMEMO encryption

OMEMO is an extension to XMPP for multi-client end-to-end encryption. OMEMO only works if the fingerprint of your and your friend's device match. To compare them, open one of your conversations and click on your profile picture next to one of your messages. At the same time, your friend clicks on your icon on his phone.

Now, both of you should see a fingerprint that you can check. If they match, change the slider as you see in the screenshot to the right.

If OMEMO cannot be activated, just send a message in the chat window. This sometimes helps. Also, it may help to end a conversation by pressing the menu on the top right inside a conversation, and then re-open the conversation again.

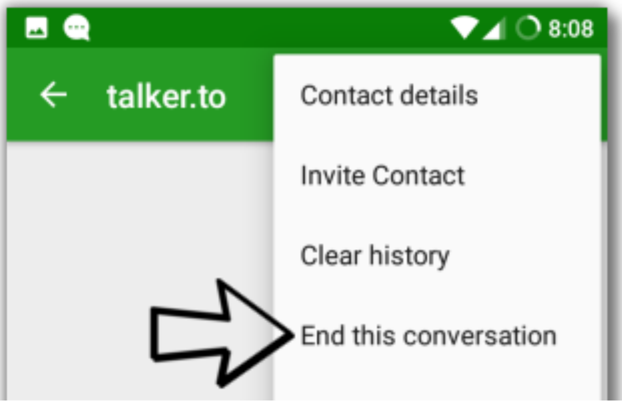
After you activated OMEMO, the input field at the bottom should say you can now send encrypted messages:



The shield or padlock indicates an encrypted message

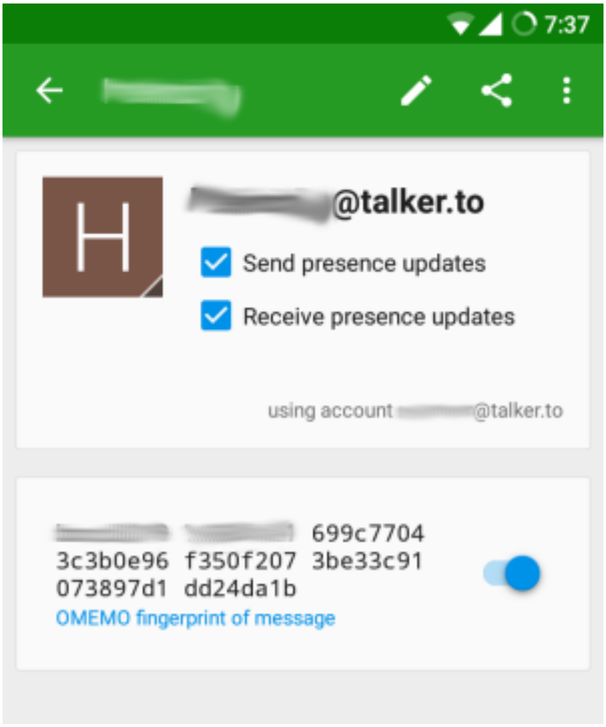
Troubleshooting Conversations

If OMEMO cannot be activated, just send a message in the chat window. This sometimes helps. Also, it may help to end a conversation by pressing the menu on the top right inside a conversation as shown in the following screenshot, and then re-open the conversation again.



Ending a conversation

Allow presence updates, this is used by OMEMO to exchange keys: In a conversation, click on the icon/image of your chat partner. In the new screen (as shown below)], make sure that all checkboxes are activated:



[redacted]

[redacted]@talker.to

[✓] Send presence updates

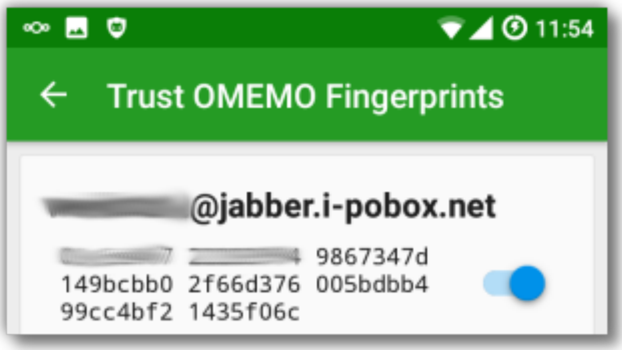
[✓] Receive presence updates

using account [redacted]@talker.to

```
[redacted] [redacted] 699c7704
3c3b0e96 f350f207 3be33c91
073897d1 dd24da1b
```
OMEMO fingerprint of message

Make sure you allow presence updates so your client can exchange OMEMO keys

Check fingerprints: You might be asked to trust fingerprints like this:


11:54

Trust OMEMO Fingerprints

[redacted]@jabber.i-pobox.net

```
[redacted] [redacted] 9867347d
149bcbb0 2f66d376 005bdbb4
99cc4bf2 1435f06c
```

Checking fingerprints

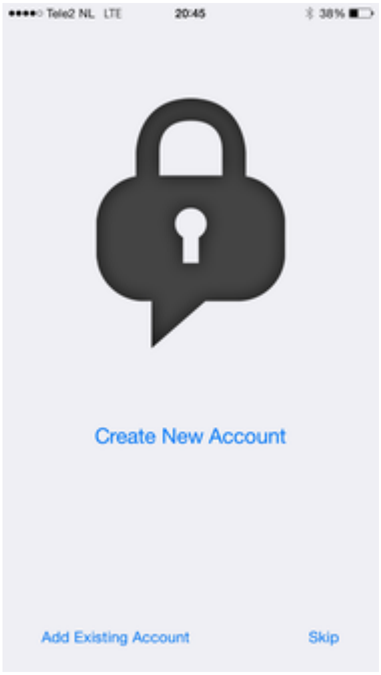
If you run into problems try asking for help in the Conversations XMPP groupchat: conversations@conference.siacs.eu

ChatSecure, iOS

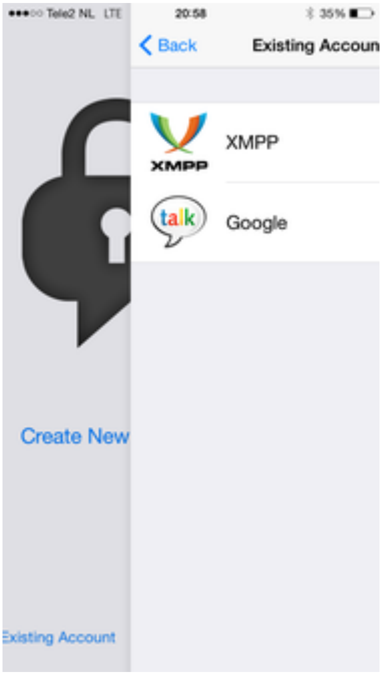
Get ChatSecure from the AppStore. ¯_(ツ)_/¯

- Start the messenger app and register / log in -

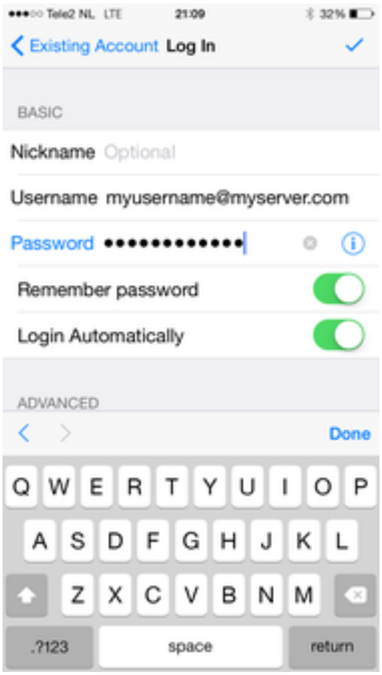
Choose whether to create a new account or login with an existing one:



Initial screen: create or add account!


●●●○○ Tele2 NL LTE
20:58
35%
Back
Existing Accoun
XMPP
XMPP
talk
Google
Create New
Existing Account

Select XMPP



Nickname Optional

Username myusername@myserver.com

Password ●●●●●●●●●●●●

Remember password

Login Automatically

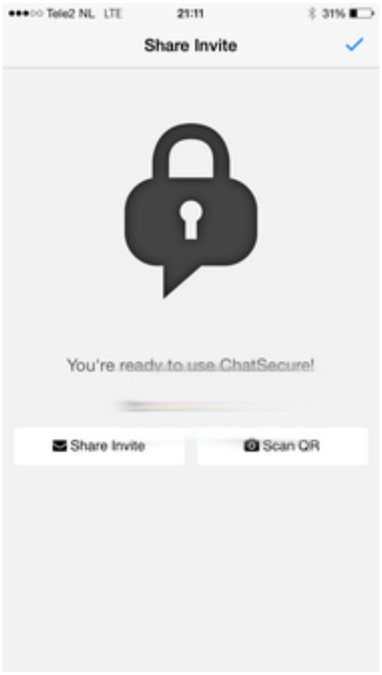
The login screen

If you already have an XMPP account, you can log in with your username@hostname and password. After you selected "Add Existing Account" you have the option to connect with "XMPP" or with "Google Talk". Select "XMPP" and fill in your Nickname, Username (username@server.net) and password. Optionally fill in the Hostname of your XMPP server and select if you want to use Tor or not. If you're doubting about the port, 5222 is the default XMPP port and would likely be on your server as well.

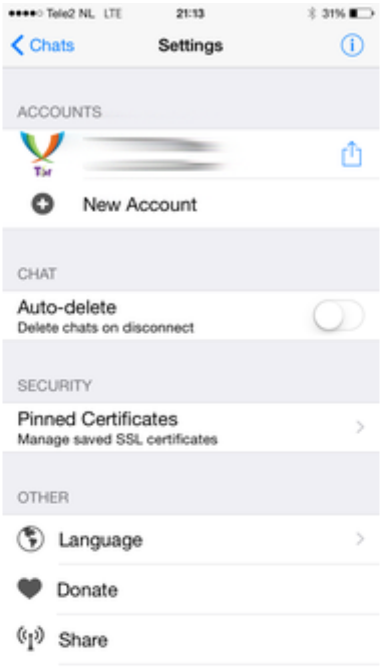
- Enabling Push -

After you've logged in, the app proposes to establish secure connections by sending an empty message to offline contacts. You have the option to "Enable push" or "skip" this part. iOS typically end the connection when an app runs in the background and requires use of Apple's Push servers to wake up and receive a message. By sending empty messages ChatSecure limits the data being sent to the Apple Cloud's Push Server but obviously still provide their vertically integrated cloud platform with meta-data. Read more about the Push issues here[16] and here[17].

n the next screen you can "Share invite" (let people on social media know about the app) or tap the '✓' symbol in the top right corner to continue. This takes you to the general 'Settings' menu.



Invite others to use ChatSecure



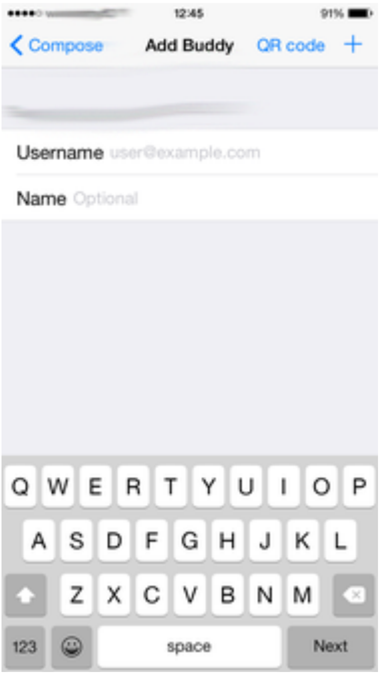
Settings

If you are successfully connected, the word "Connected" appears right under your username. Before you can edit your account settings, you need to log out. To do this, click your account/ nickname in the settings menu and select "Log Out".

- Create New Account -

Choose "Create New Account" and give your preferred nickname. Under "show advanced options" you can customize your username, generate an automatic password, enable TOR (we didn't test it)] and select a server where you would like to register your account on. This is the server you will use to communicate with other people's selected servers, and depending on the server settings it will also store your (encrypted)] messages. ChatSecure let's you choose between 3 built-in servers options. Default is DuckDuckGo, but when you tap on "DuckDuckGo" the app will take you to the server selection screen where you can choose between DuckDuckGo, Calyxinstitute.org and OTR.im[ref]All three of these servers score poorly on the modern XMPP compliance test[18], it also offers you the option to select another, custom, server. Here you can fill in the hostname of the XMPP server of a friend.

- Adding contacts -



Friend request

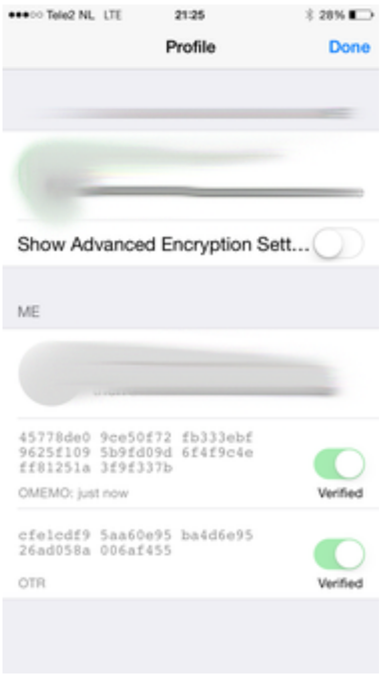
From the settings menu, tap 'Chats' (top left)] to start chatting and adding friends. To add friends tap the 'Compose' icon, top left corner. Then tap "Add Buddy" and fill in your friends username and hostname (username@hostname) or scan their QR code.

Click the "+" icon when you are ready. Your friend will now appear in the "Chats" list and will be available for conversation after being approved by the other side ("pending approval"). After this, tap your friends name to start chatting.

If you get a friend request, their nickname will appear in the "Chats" list.

- Encryption -

When in a chat, tap the information icon on the top right (i) to change your encryption settings. The information menu displays your current and past verified fingerprints and allows you to specify an encryption method by tapping "Show Advanced Encryption Sett...".



●●●○○ Tele2 NL LTE 21:25 28%

Profile Done

[redacted]

[redacted]

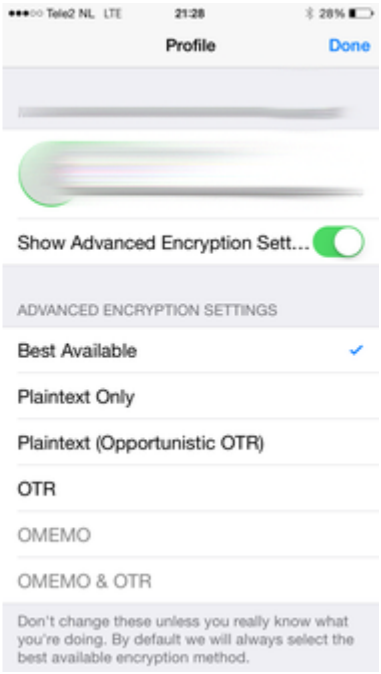
Show Advanced Encryption Sett... [ ]

ME

[redacted]

```
45778de0 9ce50f72 fb333ebf
9625f109 5b9fd09d 6f4f9c4e
ff81251a 3f9f337b
```
OMEMO: just now — Verified [✓]

```
cfe1cdf9 5aa60e95 ba4d6e95
26ad058a 006af455
```
OTR — Verified [✓]



●●●○○ Tele2 NL LTE 21:28 28%

Profile Done

[redacted]

[redacted]

Show Advanced Encryption Sett... [✓]

ADVANCED ENCRYPTION SETTINGS

[✓ Best Available]

Plaintext Only

Plaintext (Opportunistic OTR)

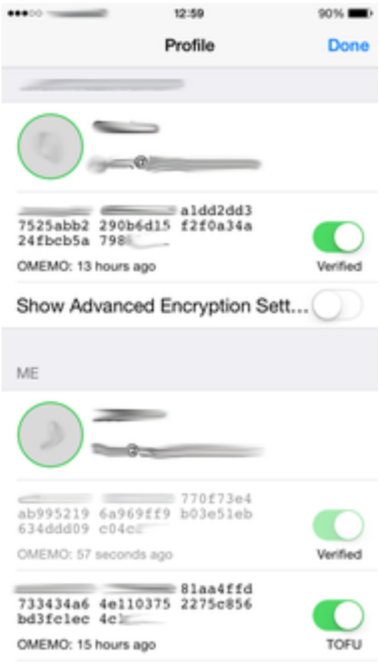
OTR

OMEMO

OMEMO & OTR

Don't change these unless you really know what you're doing. By default we will always select the best available encryption method.

At the time of writing OMEMO works well with other OMEMO clients, images shared over HTTPUpload however are not displayed inline but rather as a URL. If you click that your browser will open it and fail to decrypt the OMEMO encoded image, because it has no notion of your OMEMO fingerprints. So for now the images shared over HTTPUpload have to be shared using plaintext.
ChatSecure implements OMEMO and OTR on a TOFU or “trust on first use" basis. New "buddies" are automatically trusted.



[redacted]

[redacted]
[redacted]

[redacted] [redacted] a1dd2dd3
7525abb2 290b6d15 f2f0a34a
24fbcb5a 798[redacted]
OMEMO: 13 hours ago
Verified

Show Advanced Encryption Sett...

ME

[redacted]
[redacted]

[redacted] [redacted] 770f73e4
ab995219 6a969ff9 b03e51eb
634ddd09 c04c[redacted]
OMEMO: 57 seconds ago
Verified

[redacted] [redacted] 81aa4ffd
733434a6 4e110375 2275c856
bd3fc1ec 4c[redacted]
OMEMO: 15 hours ago
TOFU

You can also untrust your friends devices/fingerprints by sliding the green "Verified" button and share fingerprints by tapping them and selecting a medium to share your fingerprint over.

If OMEMO cannot be activated, just send a message in the chat window. This sometimes helps. Also, it may help to relaunch the app...If you're chatting with someone using something else than ChatSecure, for example Conversations on Android it helps when the Android side allows for receiving and sending presence updates. For specifics refer to the Conversations section of this guide.

Gajim, Desktop Windows / Linux

These instructions are for Debian / Linux. For Windows it is possible to download the binaries[19].

- Getting the latest version of Gajim -

The version that is packaged in the repositories of Debian does not support OMEMO unfortunately. As a way around, you can download and install the latest version of Gajim from the Debian backports repositories.

In case you don't have backports on your sources.list, follow these instructions before you start:

For wheezy add this line to your sources.list (or add a new file with the ".list" extension to /etc/apt/sources.list.d/) You can also find a list of other mirrors at https://www.debian.org/mirror/list:

```
deb http://ftp.debian.org/debian wheezy-backports main
```

For jessie add this line to your sources.list:

```
deb http://ftp.debian.org/debian jessie-backports main
```

Afterwards

```
sudo apt-get update
```

Now we are ready to go!

- Installing Gajim & other dependencies from backports -

To install gajim:

```
apt-get -t jessie-backports install gajim
```

Now you'll also need to install Python-axolotl, which will allow you to setup a security layer on top of XMPP. Run:

```
apt-get install python-axolotl
```

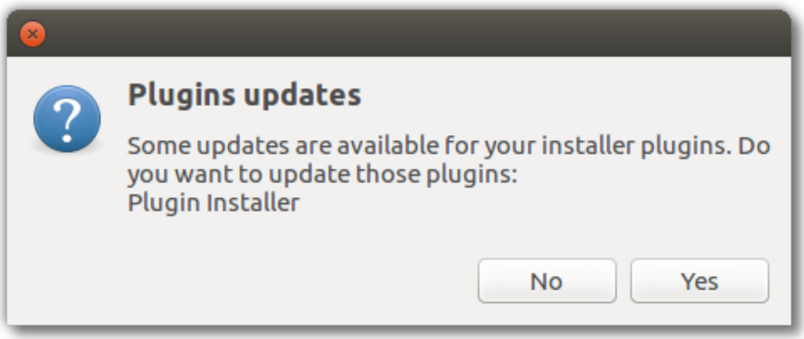
Next, you have to downgrade protobuf due to a bug in python-axolotl:

sudo pip2 install protobuf==2.6.1 And now for OMEMO! There is a package gajim-omemo on Debian Backports. So run:

```
apt-get -t jessie-backports install gajim-omemo
```

- Starting Gajim and installing plugins -

Next, start Gajim. After Gajim has started, wait some seconds until it requests your permission to install updates:


Plugins updates

Some updates are available for your installer plugins. Do you want to update those plugins:
Plugin Installer

Allow Gajim to update itself

Allow this. Afterwards, a new window will open that lists all components that can be installed and updated. In this list, activate the checkbox next to the following plugins:

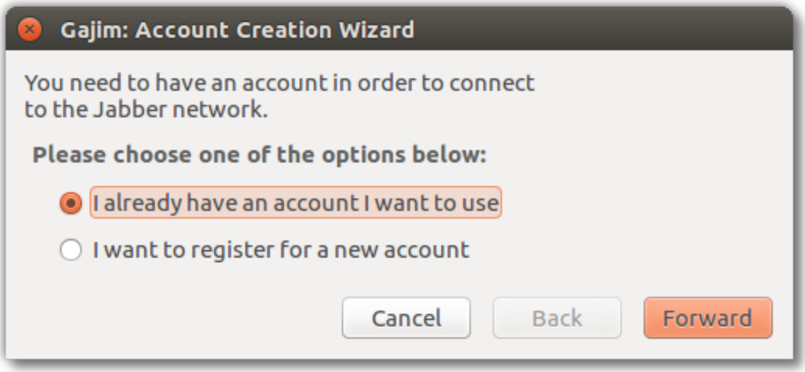
- OMEMO
- HttpUpload
- Image
- Url image preview

These plugins allow for encryption (OMEMO) and the easy sharing and display of images across different clients (HttpUpload, Image, Url Image preview)

Then, click the button *Install/Upgrade* on the bottom left on that window.

After the update has finished, go to the other tab *Installed*. There, make sure that all components are activated via the checkbox. Afterwards, click close on the bottom right of the window.

Then, you should see a wizard to setup your XMPP account. Select the option that you already have an account and follow all instructions yourself using the default settings.


Gajim: Account Creation Wizard

You need to have an account in order to connect to the Jabber network.

Please choose one of the options below:

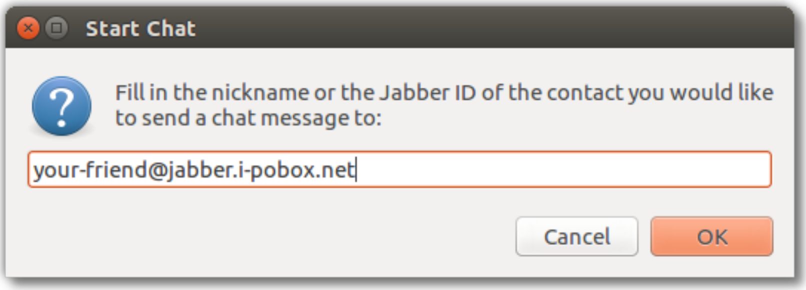
[✓ I already have an account I want to use]

I want to register for a new account

Gajim account creation wizard

After you finished the wizard successfully, Gajim will show your status as Available. Congratulations! Now, let's send messages to your friends.

To do so, click on the Gajim window and move your mouse to the top of the screen. There, a menu should appear. Go to Actions -> Start chat… . In the new window, add the XMPP ID of your friend and click ok.


Start Chat

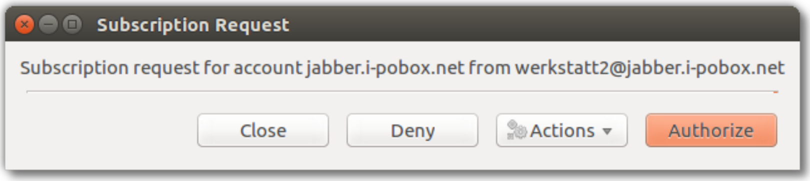
Fill in the nickname or the Jabber ID of the contact you would like to send a chat message to:

your-friend@jabber.i-pobox.net

Adding contacts

Go to the main menu again and select View -> Show offline contacts… . In the Gajim window, you should see your friend. Right click on the name of your friend and select Manage contact -> Add to roster. In the pop up, just click Add. Now your friend is permanently added to your list of contacts. Next, right click on your friend and select Manage contact -> Allow subscription -> Allow contact to see my status.

Your friend should see a request like this:



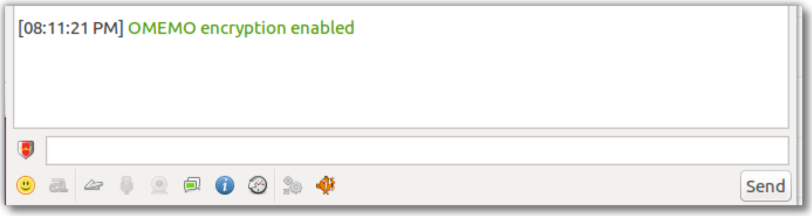
Subscription request for account jabber.i-pobox.net from werkstatt2@jabber.i-pobox.net

Friend request

Your friend should click *Authorize*, which enables her to see if you are online or not. Also, this step is necessary for activating the encryption.

Next, make sure that your friend also allows you to see her status.

Now, when you open the chat window to your friend, it should say OMEMO encryption enabled and show a red shield next to the input field, like this:



[08:11:21 PM] OMEMO encryption enabled

Omemo enabled

If you don’t see the OMEMO encryption enabled—just restart Gajim and have a look again.

You might at some point be confronted with a window about trusting fingerprints.

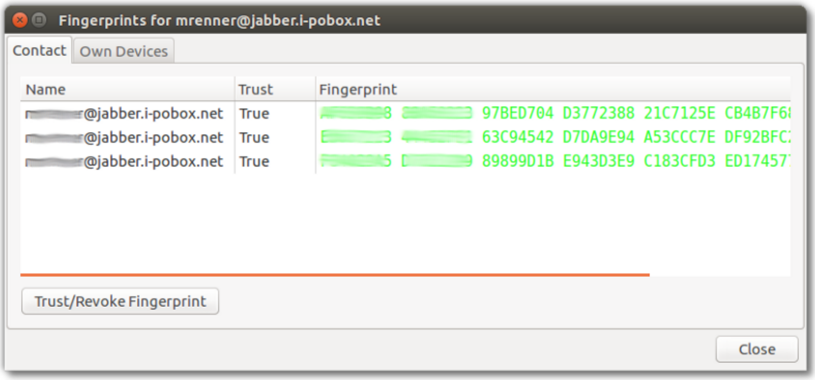
- Fingerprints -

Simply put, a fingerprint is an ID of a device someone uses for the messaging. In order to make sure that you communicate with exact the devices, which your friend uses, you need to see if the fingerprints listed in this window match with the ones your friend really has.

So, ask your friend to list her fingerprints on her desktop. On her computer, in the chat window with you, she should click on the setting symbol below the text input field (grey, with wheels)]. From there to *OMEMO encryption -> Fingerprints*. She should now see the same window as you.

She should chose the tab Own devices, while you chose the tab Contact. Now, select a fingerprint that matches with the one of your friend and press the button Trust/Revoke Fingerprint. Also press yes in the window that appears.

Finally, all fingerprints should be green like this:



Fingerprints for mrenner@jabber.i-pobox.net

Name	Trust	Fingerprint
[redacted]@jabber.i-pobox.net	True	[redacted]8 [redacted] 97BED704 D3772388 21C7125E CB4B7F6
[redacted]@jabber.i-pobox.net	True	[redacted]3 [redacted] 63C94542 D7DA9E94 A53CCC7E DF92BFC
[redacted]@jabber.i-pobox.net	True	[redacted]5 [redacted]9 89899D1B E943D3E9 C183CFD3 ED17457

Omemo enabled

- Troubleshooting -

Sometimes, a restart of Gajim just helps :)

If OMEMO encryption or the fingerprint option is grey and cannot be activated, just send a message in the chat window. This sometimes helps.

If you wish to know more about Gajim check out the documentation[20]. For more advanced issues check out Gajim's XMPP chatroom[21].

1: https://www.mathias-renner.com/
2: https://bitleaf.de/2016/11/28/setup-a-whatsapp-like-chat-messaging-that-respects-your-privacy-in-just-10-minutes/
3: https://bitleaf.de/2016/12/07/setup-xmpp-with-omemo-encryption-on-your-desktop/
4: https://xmpp.net/directory.php
5: https://compliance.conversations.im/
6: https://github.com/iNPUTmice/ComplianceTester
7: https://conversations.im/
8: https://xmpp.org/software/clients.html
9: http://omemo.top/
10: https://xmpp.org/extensions/xep-0163.html
11: https://xmpp.org/extensions/xep-0198.html
12: https://xmpp.org/extensions/xep-0280.html
13: https://xmpp.org/extensions/xep-0313.html
14: https://xmpp.org/extensions/xep-0363.html
15: https://f-droid.org
16: https://chatsecure.org/blog/chatsecure-v32-push/
17: https://chatsecure.org/blog/fixing-the-xmpp-push-problem/
18: https://compliance.conversations.im/
19: https://gajim.org/downloads.php?lang=en#windows
20: https://dev.gajim.org/gajim/gajim/wikis/help/home
21: xmpp://gajim@conference.gajim.org/?join

Beginner's Guide to XMPP Speak

homebrewserver.club

XMPP

Extensible Messaging and Presence Protocol[1] - A communications protocol based on XML that has been in development since 1999. Ever since the standard has been incrementally developed to add more functionality. It is the underlying technology that powers a lot of well known chat applications such as WhatsApp and Google Talk.

Jabber - The original trademarked name of the Jabber service. Jabber.org[2] is the original instant messaging (IM) service based on XMPP. Afterwards many different servers and clients have emerged. "Jabber" is to "XMPP", what "email" is to "SMTP" and what "web" is to "HTTP".

MUC - "Multi-User Chat", the jargon for groupchat in XMPP world. This feature needs to be supported by both the clients and the servers.

ROSTER - is your list of contacts.

JID - Jabber ID / XMPP address. JID is the identifier of a user account. It looks a lot like an email address: user@server.com, but it is not. Some users might use the same name for both their email and JID but most of the time these are completely different things. Following the same logic, chatrooms also have a similar address: roomname@muc.server.com.

XEP - XMPP Extension Protocol

XEP-0045: MUC[3] - defines support for Multi-User Chats, in other words, group chats.

XEP-0163: PEP[4] - Personal Eventing Protocol allows amongst other things to automatically publish avatars and OMEMO public keys.

XEP-0313: MAM[5] - Message Archive Management is an extension that allows one to receive messages while offline.

Software: Clients

As featured in our guide on XMPP clients:

Conversations[6] - Mobile client for Android.

Gajim[7] - Desktop client for Linux distributions, BSD, and Windows.

ChatSecure[8] - Mobile client for Apple iOS, 'experimental', but in active development.

Other popular clients not featured in our guide:

Adium[9] - Desktop client for Apple macOS. The OSX version of Pidgin

Kaiwa[10] - A webclient, so it runs in the browser. Supports a lot of features and XEPs. Win/Lin/OSX

Pidgin[11] - A client which supports a number of messaging standards including XMPP. Recently implemented support for OMEMO.

There are many more clients available[12]. Check your local F-Droid/ Google Play Store/AppStore.

Software: Server

Federated server - A group of servers which agreed upon certain standards to communicate with each other. Such a group is a federation of servers. The federated XMPP protocol enables the user to select a client of preference and connect to their XMPP server of choice.

Centralized service - A vertically integrated service that includes both exclusive client and server software. In this scenario, most of the time, the user can only run one specific client and only interact with other users from the same service.

Prosody[13] - Open Source XMPP Server software written in LUA[14]. It is actively being developed and is notable for the large ammount of supported XEPs[15].

Encryption

C2S - The connection between a client and the server.

S2S - The connection between servers.

Transport Layer encryption - Encrypts communication while it is in transit between client and server (c2s) or from one server to another (s2s). The servers where the messages are relayed between can however still read their content. It is probably known to most people in the form of HTTPS, which indicates the communication is encrypted between your browser and the server that is hosting the website you visit.
Depending on your threat model, in case you and your contacts share the same trusted XMPP server, transport layer encryption might be enough to safeguard your privacy.

End-To-End Encryption (e2e) - End-to-end ciphers is client side method for encrypting messages. Only the sender, and the receiver, at both ends of the communication chain, can read the message, but not the servers in between.

OTR - "Off-The-Record" is one of the older forms of e2e encryption available in some messaging clients. The big disadvantage of OTR is that both clients need to be online at the same time for the encrypted session to work. It is also not possible to synchronize OTR encrypted messages across mutliple clients.

OMEMO - OMEMO Multi-End Message and Object Encryption, OMEMO is the XMPP implementation of the Double Ratchett encryption algorithm developed for Signal by Moxie Marlinspike at Open Whisper Systems. It is the most modern and convenient encryption mechanism that is practically invisible to the user. It also provides so-called forward secrecy, which means that every message is separatly encrypted. In the case that one cipher is intercepted by a third party, only one message can thus be decrypted.

TOFU - Trust On First Use. A mechanism where the received fingerprint is assumed trusted immediately and is therefore checked as verified. Used in ChatSecure for OTR and OMEMO, called 'Blind Trust' in Conversations.

OpenPGP - Pretty good Privacy is the oldest generic method for end-to-end encryption. It requires quite some knowledge and maintenance from its users. OMEMO is designed to provide similar or better encryption with less hassle. To use OpenPGP in Conversation a third party app called OpenKeyChain is required.

Threat Model - When thinking about security and privacy it is important to note that there is no such thing as a protection against every and any possible threats. By aiming too large and aimlessly at a universal form of privacy, there is a risk of missing obvious blind spots because of lack of resources, lack of time, and lack of knowledge to cover all possible situations. In that sense the concept of a *threat model* is very useful. In a threat model, an assessment of what has to be secured and who could be willing to acquire your information and at what cost, is established in a realistic fashion. What poses a credible threat to you and your situation? Who represents that threat? What kind of resources does this threat possesses? The answer to these questions should inform you on what kind of measures one should take and which ones have the highest priority.
Obviously this differs from situation to situation. Are you a political activists or dissident trying to organise for direct action and trying to avoid surveillance from government agencies? Are you the user of a popular social network, trying to protect as much as possible your most confidential information from your private life? Are you an office worker trying to leak confidential information about unethical activities of your employer while remaining anonymous? Are you a user of public or private torrent trackers hoping to get away with mass downloading and uploading of whole seasons of The Great British Baking Show? Every situation is different, every situation needs a specific understanding of what is at stake and what would be the consequence if what you try to protect is exposed. Don't believe in magical solutions, do your homework.

This list is partly based on this glossary[16].

1: https://xmpp.org/about/

2: https://en.wikipedia.org/wiki/Jabber.org

3: https://xmpp.org/extensions/xep-0045.html

4: https://xmpp.org/extensions/xep-0163.html

5: https://xmpp.org/extensions/xep-0313.html

6: https://conversations.im/

7: https://gajim.org/

8: https://chatsecure.org/

9: https://adium.im/

10: http://getkaiwa.com/
11: https://www.pidgin.im/about/
12: https://xmpp.org/software/clients.html
13: https://prosody.im/
14: https://www.lua.org/about.html
15: https://prosody.im/doc/xeplist
16: https://wiki.xmpp.org/web/Usability/Glossary