Discussion

Motivation

Today, most users access email via big service providers, such as Google, Yahoo and others. In 2017, 83.5% of 14 to 18 year olds reported Gmail as their primary email provider (source). In our view, there are several things that are wrong with this picture:

- Big tech companies make money by selling their customer's information to advertisers. You are not their customer, you are their product.
- Your online identity does not belong to you, and it can be suspended or terminated at any time. It does happen.
- On a more idealistic note, the Internet was supposed to be a set of open standards to connect individuals and organizations. When did email become so complicated that we need trillion dollar companies to help us to send messages to each other? The ability to exchange messages in a secure way is just too fundamental to hand it over to a third party.

This project is about making email simple, secure, decentralized and free from government or corporate censorship. To achieve this, we need to take user identity back to the user, which will have impact far beyond just email.

It All Starts with Identity

Your identity controls what you can do online. For most users it includes logins to multiple services, such as Google, Facebook, Amazon, etc. The commercial companies controlling such services have full control over your identity - they can restrict or disable it at any time, without explanation or recourse.

The identity provided by such services also denies you privacy - their business model is to sell your private information to advertisers. Sure we can argue if their service is worth it, but arguably, for most of us, the answer to the question "how much of your privacy would you like to give up", the answer is, "none".

Our goal is to change this state of affairs, and give control over the user identity back to the user. It can then be used for authorization, secure communication, or to delegate some limited authority to act on the user's behalf to a third party.

Decentralized Identity Registry

Let's imagine instead a system where you control your identity, which means that you can prove to anyone that certain data originates from you. A common way to do it would be to have a private key and use it to sign outgoing messages. It can be also used to derive unique encryption keys so only the intended recipient of a message can read it.

Now let's assume that there is a registry where all the public keys are stored. Each key can be associated with a unique name. Whoever controls the corresponding private key, also controls the name.

This would give you something like a global user name - instead of having a single name and password per service, you can use your private key to control them all.

Email

Which brings us to email, the biggest messaging system out there. It was supposed to be an easy way to send messages between users, and initially it was. However, the simplicity had its downside - because anyone could send messages to anyone else, without authentication or encryption, the abuse became common, and to confront this abuse, a patchwork of technologies was applied, making email increasingly more complex and unwieldy.

It's not that you can't run your own email service. You sure can. You will be joining the elite community of about five people who do that. In case you are interested, here is how: Part 1, Part 2, Part 3, Part 4. Good luck.

Instead, we could use encryption and the identity registry to send emails between users in our system in a way that ensures privacy and prevents spam and abuse. Here, we have a working system that demonstrates this concept.

Architecture

For this prototype, we use the following architecture:

- *Identity service*, which is tasked with registering private keys, unique names and addresses
- *Dump service*, which accepts any valid messages address to any users, and stores them until the recipient comes to retrieve them.
- *Email client proxy* which allows any email client to retrieve and send messages. The proxy handles the identity verification and encryption.

If you want convenience, you are free to use public servers that we run. Notice that all of them, with the exception of proxy server, handle encrypted messages, so for them they are just meaningless bytes. The proxy server is different - since it talks to the end-user email client, it must be able to decrypt messages. If it does sound scary, you are right. But here are a few things to make it a little less scary:

- The proxy server does not store decrypted messages anywhere (except in memory, temporary, before it sends them to the client).
- You must send user information to the proxy so it can construct your private key used to retrieve and decrypt messages. Again, the private key is not stored anywhere, except in memory, for the duration of the user session.
- If you are still uncomfortable with your private key being sent over to the service, you were paying attention, congrats. To make you feel a little

better, the key you sent over can be made limited in scope to just encrypt and decrypt messages. Another key (parent key) will be responsible for address and name assignment, and it can disable the child key at any time. The parent key never leaves your hands. If you suspect that your child key became compromised, just use the parent key to disable it, generate another child key and reassign the name to it.

- In case you are absolutely against sending any private keys over, you don't have to - just run the proxy server locally on your machine, or on your cloud provider VM, or whatever.

Aside - the "at" addresses

Every email address comes in bob@server format. Now that we talk about global identity, the "@server" part becomes obsolete - and it's no surprise that many services just use the username, or @bob handle. We have to comply with the email address format, but we want to de-emphasize the "at" part, so we will use the simplest possible format, bob@x. Yes, this is the user name followed by the "at" symbol and the letter x. We use x because it's cool, and because it looks like we just cancelled the whole service part - we put a cross in its place.

Unfortunately some email clients might not be happy with "@x", in this case you can use "@ubikom.cc".