

User Chat Expectations

Posted on December 26, 2013

This post was developed as part of a digression for a another post coming next.

Right now, one of my core focuses revolve around making a better chat service. I also find it to be one of the best use cases for maturing the event distribution system I am trying to design.

Related work

Lately—due to US government being revealed to spy, and with companies like Microsoft, Google, Facebook, et al. willingly or unwillingly cooperating with giving information out—groups have been rapidly trying to find or create alternatives for secure, private communication.

[Skype](#) is one of the leading chat services now in both personal and business communications. Microsoft [bought](#) Skype in 2011. Since then, they've changed the service in ways that seem to allow for surveillance.

Many attempts like [RetroShare](#) go for complete security, but without portability. These non portable attempts usually use a [DHT][] as a mechanism to facilitate connecting up with others. However, it tries to do everything and anything it can to make it so you don't need another service. I find that the user experience of this application is not acceptable, especially for normal users. RetroShare is made in C++.

Then there's [Aether](#) which is meant to facilitate discussion like a forum, rather than a chat mechanism. All use is meant to be anonymous; it is distributed and encrypted. The user experience is alright, but the program is incredibly buggy and breaks a lot. Aether is made in Python and JavaScript.

The one that impressed me the most is [Tox](#). This project focuses on a pleasing user experience

Tox explicitly aims to replace Skype for those that care. They go for a UDP delivery system, rather than TCP. Tox uses a DHT for coordination, and supports single and group chats, as well as transmission of files and voice or video transmission. I am not sure how this service will perform on a mobile platform. Tox is made in C.

There are other services as well, but they have not caught my eye. Most of them are now aiming for complete privacy and independence from any networks, in opposition to NSA efforts. These projects are often open source to provide comfort in the users that there's no tricks behind the services. However, my opinion is that many of these projects are done without guidance from an experienced and / or educated cryptography domain specialist.

Intentions

Now that I have documented some other works that seem interesting, what am I planning? What is the general goal that I have before we continue and design something?

Disclaimer

I would like to [learn to crypto](#), but my time is limited.

I'd rather not run into where I am ashamed like [Adobe's incident](#) because I did something seriously ignorant. *Encrypting with the same key instead of just hashing really ought to have made an expert cry vicious tears.*

I do have some experience to the point of not make mistakes in web services like..

■ Or, you used Encrypt-and-MAC instead of Encrypt-then-MAC you dummy!

Plan

The goal is to have a service which can work on its own like in a desktop application, as a footer chat like [cometchat](#), and have no difficulty with a mobile environment.

Since each site can have its own community, naturally, each community defines the relations each user has with each other. Or even how relations might be established. However, it will not be self-hosted like how cometchat is made to be.

Where this is unique

A big problem that I see today is that many sites go for services like cometchat, which cannot be served effectively outside of the site in question. Users end up sharing their contact details due to the frustrations of using the service for reasons like needing other common features—such as file transmission. If such a service as what I described were integrated with sites, then a user could send a request directly to add the other user to their global contacts list.

The consequences of allowing such integration means that users which have established their presence with such sites can also access their per-site contacts list (if that site provides the services to allow it) on the stand-alone application.

Recap

The service is to fulfill these principles, which are ordered by priority:

1. Support desktop, mobile, and web platforms.
2. Easy integration with other sites.
3. Privacy
4. Extensibility for future change

It does seem like privacy is low on that list, though I do believe it to be possible, even in situations like running in a javascript environment. *So long as the hosting environment on the web can be trusted too for what goes on within the page.*

Definitions

First, let's define the essential component of a chat session: a conversation.

Each chat session is made up of successive events that describe messages that users send to one another.

In other words, the chat experience is a set of chat sessions which is composed of a sequence of message events.

Yet, chat sessions can also vary in how participants interact. A few forms that we'll look at are:

- One on One
- A Group Conversation / Session

However group conversations, or rather established unique chat sessions, can have very different rules depending on the implementation.

Comparison

In this section, we'll compare IRC, classic MSN, and Skype for how things work in a group chat session. This comparison will point out what features are present and what features might be considered lacking.

IRC

IRC has post-active moderating features, meaning that it can ban users from channels based on given criteria (such as an IP mask, or name). But it cannot retract or edit information that has already been sent. There were established masters, and moderators, which were varying levels of *Operator* titles. Each channel or established group chat session. Each channel can have various configurations applied, such as a title, and whether or not identified users can post or not. You cannot sign in twice under the same user identity directly. If you connect, no past messages are sent to you.

Classic MSN

Note, my comments on classic MSN are likely inaccurate. For comparison only.

MSN group chats were short lived, though they are logged for searching after the fact. Users could edit their own posts in the past within the last minute. Users could not edit others post, as there was no established master. Older messages that were not sent to another client under your user, will be sent to you upon reconnection.

MSN has been merged into the Skype network these days.

Skype

Skype does have moderating features, such as deleting or editing messages. Masters in a group chat session may edit or remove posts by others. Users may also edit or delete their posts, usually within the time limit of 5 minutes. Skype also logs messages locally. You may only search what happens to be on your screen with `Ctrl + F`, though there's a unified search which can query the message histories. You can attempt to expand what is shown on the screen by scrolling up, which gets some time interval of messages. There are no public chats like on IRC, so joining a chat is completely by invitation by someone already inside. Group chats are initiated by bringing a third user into a one-on-one conversation. Invitations involuntarily bring the invited user into the group conversation.

Time to make decisions



So, we have some decisions to make. What will we take and what will we put aside? Where will we innovate? Let's see what the target users expect.

With several targeted interviews we find that the users expect..

- Privacy in..
 - Messages
 - Profile details
 - Current Status
- Log for conversations
- Group conversation invites should be treated act similar to contact requests. That is, it requires explicit action, rather than by force.
- Per-Conversation notification options which also work on mobile. These notification options are shared amongst clients as well.
- Editing past messages
- Deleting / redacting past messages
- A desktop application (for at least OS X and Windows)
- A mobile application
 - iOS
 - Android
 - Windows Phone

What users expect, but did not mention...

- No forgery of messages
- No man-in-the-middle manipulation of messages

Next, we have some things to consider in terms of innovation. We consider seamlessly integrating with site communities a pivotal innovation that could set us apart and rapidly gain a user base.

To support seamless integration with other sites, this service should come with features such as:

- Web tool-bar or other add-on representations
 - Can notify when messages come from other contexts.
- Site-specific public chat sessions
- Page-specific chat contexts
- Site-specific context that can be accessed from non-browser environments. (*Desktop, mobile, etc.*)
 - Remote invalidation for when the user changes their credentials
 - Multiple *named* contexts available on each site.