Taurus Project Evaluation Report

First and foremost, I hope not to be this far down to the wire on a software/documentation project again in the foreseeable future!

The biggest challenge on this project was reaching the design stage so late. I intentionally put off any major implementation decisions until after we'd had a chance to discuss design, which left nearly everything until the last minute. I even had to simplify from my initial design sketch (preserved in doc/notes), although I take full responsibility for getting sidetracked by extra features like the curses interface. (I'm not sorry about this either,

Because I knew I was going to need the CipherSaber2 encryption regardless, I was able to start that a bit earlier, which is why it's so much better-represented in the test plan. In retrospect, I might also have been able to get away with starting the listening daemon earlier, at least after deciding it would be its own separate piece. As it is, I ended up having to revise it when I changed from single-message files to conversation files, and it wasn't too much trouble.

The correspondant-based conversation files were one of the better ideas I had in designing this project. It's the natural way to view a sequence of messages between individuals, and is consistent with nearly every other interface for doing so. Storing them that way avoided having to write a lot of logic to create that display, or else provide a lousy message-by-message user interface. (It turns out that tail -f in python is pretty easy with generators, who knew?)

That said, I can still imagine implementing the "message blob" idea from my earlier design in a later version of Taurus. It's more complex to interact with, but has a number of security advantages which are detailed in the early design draft. Alternatively, an actual database would be much easier to interact with and could probably provide some of the same security advantages; I haven't looked into this. A database would also be hugely advantageous in handling concurrency, since it's designed to be accessed concurrently.

The most conspicuous feature I wanted to implement and didn't make time for was managing the daemon from inside the client. The second most conspicuous is a smoother setup process (although at least I documented what was required fairly well). There are a huge pile of open issues in the repo right now (24, compared to 6 closed ones), most of which are minor features or refactoring. The code's very messy right now, and significantly undertested. It'll be nice to take a breather and then clean it up into something presentable.

Last but not least, I have to admit … as stressful as it was to get all these documents written, especially at the last minute … I found several bugs just by working on them (especially but not only the test plan).