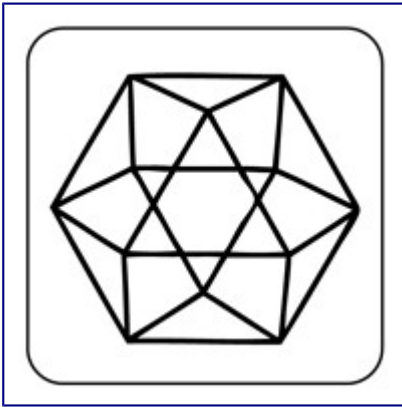


Please Stop Writing Secure Messaging Tools




Dymaxion.org

This is my first Patreon-supported essay, which went out a week early to my \$10 and up subscribers. This was written in part in response to issues with the IWMF check-in tool, but it's something I've been concerned about for some time.

If you like the work I do, you can support it via Flattr:



 I have a Patreon, [here](#), where you can subscribe to support my security and systems-focused writing. You sign up for a fixed amount per essay (with an optional monthly cap), and you'll be notified every time I publish something new. At higher support levels, you'll get early access, a chance to get in-depth answers to your questions, and even for more general consulting time.

[Return to All Essays](#)

Folks, you're doing awesome work, but enough is enough. We have enough secure messaging tools. We don't need any more.

I exaggerate. We still need better messaging tools that hide communications metadata. We need tools that can be used by nontechnical users in realistic scenarios. We need tools that let users get the full benefits of the security invariants the tools attempt to maintain. We need tools that let users move seamlessly between mobile and desktop, that let them communicate over local networks when the Internet is down, and that don't ask them to trust any central server. We need tools that do all of this and don't try to tell users how they should work, but rather support them in accomplishing their goals. This is a Hard Problem.

Tools like Trello and Slack have become popular because they've devoted a lot of polish to eliminating usability speedbumps for the tasks users are trying to accomplish in the world, not just for tasks in the

software. Users care because they're trying to get work done, not demo security tools. The risk analysis of a potential attack versus a definite work slowdown is more complex than the security community often wants to accept.

Given that we still have so far to go, why am I telling people they should stop writing secure messaging tools? Because we have too many other tools we also need. I've worked inside a number of distributed organizations and talked to folks at many more, and there's a giant list of software that people rely on to get their work done. IM, chat, and email-like systems are just one small corner of the day-to-day life of a working team, whether that's an activist group with no formal leaders or office space, an NGO with staff deployed in a half-dozen countries, a freelancer-heavy news desk, or a law firm representing surveilled clients.

People who are actively targeted by adversaries or at elevated risk, who operate in geographically dispersed teams, and who work in places with intermittent connectivity (these conditions often go together), tend to find a number of software properties useful:

- **Multi-user**, because teams are more than just two people
- **Multi-device**, because people travel and need to swap out or switch between hardware
- **Decentralized**, because leaving a server somewhere is risky, inconvenient, sometimes prohibitively expensive, and can cause latency issues
- **Client-side end-to-end encrypted**, so everything that's trusted with data is physically with someone trusted
- **Offline-friendly**, because the Internet isn't always on and people need to collaborate locally when it isn't
- **Role-aware**, because teams tend have different people doing different things, and software that doesn't support this doesn't let them work effectively; design for security means designing for community
- **Secure by design** for everything from basic communications architecture and protocol parsing through cryptographic enforcement of roles and permissions, because attackers will exploit policy weaknesses otherwise
- **Metadata-sensitive**, because adversaries don't always need content
- **Multi-organization**, because cross-organization collaboration is critical but complicates role structures and authorizations and tools that enforce silos hurt field outcomes

In some high-risk cases, folks may also need systems to be:

- **Unlinkability compatible**, so their use of the system doesn't let an adversary (or even another user) link their physical location or another identity to who they are in the system
- **Airgap-friendly**, where computation and keys can be kept on an offline device and encrypted data moved to a connected machine for synchronization, because some data is simply too dangerous to put on a connected device directly no matter what

These kinds of decentralized-first applications require a somewhat different way of looking at the world for implementers, and one of the keys is that they're intended for use within teams, not by the Internet at large. In some cases, it may be useful to provide bridges to more traditional web interfaces for interacting with folks outside a team, but that's often not relevant.

So, what do people need with this feature set that's not IM, email, or chat? A lot of things. Here's twenty-four off the top of my and a few smart people's heads:

- **Ticketing systems**, for managing everything from task assignment to emergency reporting so organizations know what's going wrong, what needs doing, who's doing it, and who's checking up on them
- **Project management and scheduling tools**, because knowing what needs doing often isn't enough if you also don't know how long it's going to take and who can work on what
- **Group password vaults**, because while password safes are great for people working alone, they're not good enough for teams
- **A Wiki/Etherpad/Word style change tracking mashup**, because while every team has documents they need everyone to have access to, some of these are sensitive, and while collaborative editing is critical some of the time, sometimes you also need changes to be examined and approved by humans with real care
- **Calendaring systems and collaborative scheduling tools**, because setting up meetings may be boring, but they're often quite sensitive and good tools can save a lot of everyone's time
- **Contacts databases**, because people's real contact information can be both sensitive and critical for an entire organization to have access to
- **Location coordination tools**, for giving teams ambient awareness of who's where without revealing that to adversaries
- **Versioned file management for media files**, because a working newsroom needs to keep track of media, but much of that media may be very sensitive before a story comes out
- **CMS and editing workflow tools**, for doing the same work on text-centric documents
- **Recording logging and proxy/full clip sharing**, for doing the same work on video or audio-centric pieces
- **Collaborative backup**, so teams can have accessible off-site backup run by people they can trust directly
- **Collaborative mind-mapping/white-boarding/presentation building systems**, for thinking collectively
- **Collaborative spreadsheets**, because they're often the best tool to deal with ad hoc problems
- **Map-based storytelling and analysis systems**, because nothing else will work for map-structured data
- **Reference management systems for papers and publications**, for long-term research projects
- **Scalable document OCR, tagging, and search systems**, because working with large document

sets collaboratively and securely is a core competency for many types of organizations

- **Library or hardware checkout management systems**, because organizations have physical resources they need to share, some of which may be very precious in their context
- **Inventory management systems**, similarly, because many organizations work on providing aid or otherwise have to manage inventories in complex, high-risk situations
- **RSS or other document queue readers with collective commenting**, for teams working together on evolving document collections
- **Stats dashboards**, because tracking performance and other scenarios is still important even in nontraditional work structures
- **Glue systems**, like bots, Yahoo Pipes, or Zapier, to pull things out of email, IM, chats, and into workflows, and to get other systems in this list talking to each other
- **Time tracking systems**, because folks need to record how much they've worked and on what, but this can be sensitive too
- **Accounting and invoicing systems**, because people need to get paid, but financial information necessarily identifies people and can involve real risks
- **Payroll and vacation tracking systems**, for similar reasons

All of these systems depend on basic communications features. Some of these needs are partially met by existing implementations, but few of those meet the architectural requirements listed above that we see in the field. When deployed together in an organization, tools like these need to share basic trust and key or authentication structures. When a team depends on multiple systems for every day work, it's critical that they share security properties and failure modes. If they don't, users will get confused about which systems can guarantee which invariants when, and problems will result. Teams may want emergency communication systems with different properties and failure modes, but normal tools need to be similar.

Guaranteeing operational similarity requires coordinating threat models and security architectures between projects. Doing this well is hard, and requires a discipline that's still rare even in the security tools community. Building tools like these right means sharing underlying libraries to do the heavy lifting for communication and security. While we may have some of the pieces here, there's not yet an obvious choice for the set communication protocols on top of which to build. My personal favorite, Briar, is designed to enable tools like these. It's not ready yet, but the Briar team got some great news recently and we should be steaming ahead soon.

In the same way that we've only learned what makes sense with secure communications tools by building them and watching people use them in the real world, we're not going to learn how to build integrated, secure, and productive decentralized work environments without getting our hands dirty. If you're considering building a communication tool that doesn't advance the state of the art for the goals I stated up front, think about whether it's likely to improve security outcomes for users. If it doesn't advance the state of the art or directly improve outcomes, consider whether your effort would be better

spent on something other than a communication tool. Diversity is good, but we have so many more things to build. Let's get started on the rest of the list.

Thanks to [@willowbl00](#), [@monstris](#), [@Oktavia](#), [@fin](#), [@drcab1e](#), and others who I'm almost certainly forgetting for input into this essay.

If you liked this essay, you can [sponsor me writing more](#). I've started a Patreon where you can pledge to support each essay I write. I'm hoping to put out one or two a month, and if I can reach my goal of having a day of writing work funded for every essay, it will make it much easier for me to find the time. In my queue right now are a piece introducing NGOs to what it takes to build good, secure software, more updates to my piece on real world use cases for high-risk users, and a multi-part series on deniability and security invariants that's been in the works for some time. I'd much rather do work that helps the community than concentrate on narrow commercial work that never sees the light of day, and you can help me do just that.

Thanks again!
Eleanor Saitta
2015.10.31
Amsterdam