Open Privacy

Research Society

The Open Privacy Research Society (Open Privacy for short) is a non-profit Canadian group based in Vancouver, British Columbia. We believe that moral systems enable consent. Our society exists to invent, create, build, test, publish, deploy, promote, and to encourage the development of such systems.

This report was authored by Research Director, **Erinn Atwater**, with additional input from **Sarah Jamie Lewis**. Illustrations by **Marcia Díaz Agudelo**.

This case study was produced in association with Privacy International and their "Building Civil Society's ability to engage on identity" project.

The goal of the "Building Civil Society's ability to engage on identity" project is to shape the discourse of digital identity whereby the interests and the needs of the individual are at the core of the design and management of any digital identity systems. Digital identity should be a tool for empowerment which enables individuals to control their data, to self-identify, and to enjoy their undamental rights with dignity and autonomy"

Introduction: LockBox

LockBox is a set of applications that arose out of Open Privacy's work helping with mutual aid funds to deploy secure infrastructure in the wake of the COVID-19 pandemic. During the March to April 2020 time frame many people, particularly those from marginalized communities found themselves in need of support.

Some community groups reached out to Open Privacy seeking advice on the need to collect sensitive information from people in these communities for the purposes of distributing donated funds. These groups were looking for better privacy guarantees for applicants and their personal information than mainstream hosted alternatives provided.

Open Privacy developed LockBox for this specific purpose: an encrypted, hosted application that allows organizations to collect information via a customizable web form. Data collected via this form is immediately encrypted using public key cryptography, and only by using a second app and a copy of the offline private key can a member of the organization decrypt and view the submitted entries.

If an attacker gets read-only access to the server where the form is hosted (such as via vulnerable co-hosted apps or weaknesses in shared host configuration), they are unable to decrypt the submitted data. Only the private key holder can decrypt submissions, and the private key file can be kept offline and only shared with people who should have access to submissions.

We worked directly with the initial group that requested our help to deploy and administer LockBox, fixing bugs and adding requested features as they arose in real time. The success and continued interest in the software lead us to promote it to a fully-supported project at Open Privacy, and we plan to release a more polished version of its two apps later in 2021, along with supporting material. Privacy International kindly funded the creation of some of these materials, as well as this case study documenting our efforts to bring radical encryption to marginalized communities and helping us to understand and plan the most useful, free, open-source end product we can create. Hopefully others will find it useful or inspiring for similar efforts, as we are a small team deploying small technology to large effect.

In what follows, we will discuss (2.1) the organization that initially requested our advice, (2.2) the needs and threat model that lead to us choosing a custom solution, (2.3) how we developed the LockBox apps and how they work, (2.4) how the organization used the software, and (2.5, 3) our conclusions and future plans for the software.

The latest copy of this document can be found at https://docs.openprivacy.ca/lockbox-case-study.

Case Study

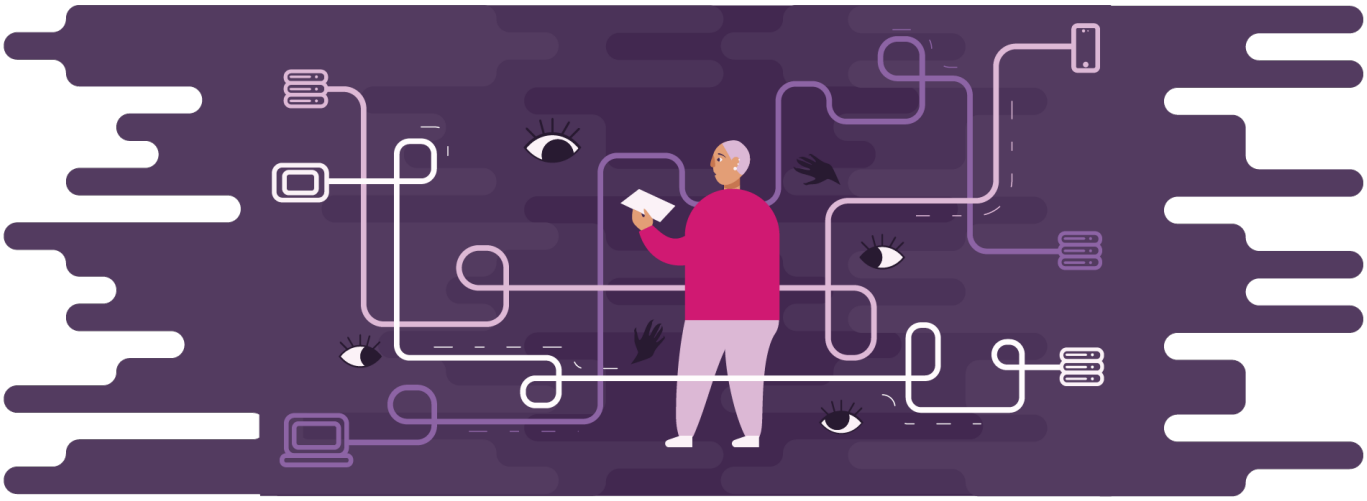
Client

Open Privacy was approached by an advocacy and support organization located in Vancouver, Canada's Downtown Eastside neighbourhood shortly after the Coronavirus pandemic began to cause lockdowns on the west coast. The organization was in the process of creating a mutual aid fund to help members who were affected financially, and sought our advice on technical platforms to use for collecting member's applications to the fund.

The organization itself is a peer-driven registered charity with approximately ten staff members, plus a varying-sized group of volunteers, with a physical location that offers drop-in services and community space for members. They do not have any staff dedicated to IT support and use Google cloud services for most of their computing needs. They do, however, have a pre-existing charitable donations platform set up, and this was used to manage *incoming* donations to the fund.

The organization's members belong to marginalized communities and have a heightened need for privacy and security of their personal information, due to the risk posed to them by society. In addition to social risks, some members face legal concerns when traveling internationally into other jurisdictions where, for instance, evidence of having previously been involved in sex work is punishable by law or extralegal measures. Thus, having all submitted data remain in Canada was also desirable. It was also pointed out that many members are distrustful of large tech companies to adequately protect their privacy.

Needs/Alternatives



Due to the heightened need for security and privacy of member data and with the potential of offering some of Open Privacy's technical staff time (thanks in large part to our independent donors!) to help set up more complex options, we initially determined that using a self-hosted solution that offered suitable encryption was within reach. As Open Privacy is a major proponent of open source software, and due to the desire for "trustworthy" options from the members the software would be serving, it was also desirable to find an open source solution.

Closed-source mainstream hosted forms such as Google Forms, although they may seem like a natural choice given the organization's existing Google Apps setup, were actually a non-option due to the organization's support of sex workers and their labour rights.

This fear is well-founded. In the United States, the Stop Enabling Sex Traffickers Act (SESTA) and Allow States and Victims to Fight Online Sex Trafficking Act (FOSTA) became law on April 11, 2018. Prior to the official signing in of the law several online services including craigslist and reddit removed part or the entire of their offerings that could be construed as relating to fostering sexual meetups. In many instances this impact went beyond removing services explicitly intended for the advertisement of sexual services. There were reports of Google locking sex workers out of files hosted on their Google Drive platform and Instagram censoring sex work related hashtags and banning the accounts of sex workers.

In what was interpreted as an act of self regulation in the wake of FOSTA-SESTA many other online platforms also revised their terms of service to prohibit adult content. Facebook banned any content that "facilitates, encourages or coordinates sexual encounters between adults", which many noted to have similarities to the language used in FOSTA. In December 2018, Jeff D'Onofrio, the CEO of Tumblr, published a blog post announcing that Tumblr was "no longer allowing adult content, including explicit sexual content and nudity."

It is important to stress that reports of online services profiling sex workers for the purposes of barring them from the service are not new, even in cases where the services being sought do not relate to sex work. Prior to both the PCEPA and FOSTA-SESTA sex workers were banned from legal fundraising efforts on GoFundMe and, at least in one case, had their medical care fundraising proceeds confiscated because of their profiled profession.

Additionally, hosted solutions generally offer only transport encryption of submitted data in order to facilitate online browser-based viewing. Some options offer symmetric key/password-based encryption, which necessitates storing the key on the hosted server and thus undermines the protection provided by using encryption in the first place. Hosted options generally do not allow much if any control or even insight into where the servers hosting the data are physically located. Self-hosting or installing a self-hosted software package on a VPS, while not a guarantee that data will never be transmitted beyond national borders, reduce the harm enough when combined with strong encryption to offer some peace of mind.

Even self-hosted and encrypted web form options, however, tend to lack public-key encryption functionality. In this configuration, the webserver accepting form submissions only has access to a public *encryption* key that *cannot* be used for *decryption*. This decryption key is stored offline for protection against potential compromises of the webserver. In the event that the server is compromised -- for example, through a vulnerability in the form app, or even a hack of another site running on the same poorly-isolated shared host -- the hacker would only get access to encrypted data.

Maintaining an offline private decryption key is an unfortunately daunting task for a web application. Web browser state is rarely "brought with" or kept around permanently by the user, and the loss of the private key is unrecoverable (resulting in the loss of all submitted data, as it cannot be decrypted even by brute force). For this reason, the approach is rarely used by web application developers who seek to maximize potential audiences. Familiar with this story-niche, Open Privacy saw an opportunity to fill a gap with the modest resources at our disposal.

LockBox development and deployment

Open Privacy assigned one developer to carry out and oversee most of the development of both apps. A second developer provided additional coding time and review, and we were fortunate to have recently hired a staff designer as well who was able to assist with the project's design, documentation and logo. We were able to leverage our open-source libraries from another project (Cwtch), including our Go-Qt cross-platform build pipeline and recently released QML widget/theme library to rapidly create a working companion application to the web app.

Prototype: Single-form, Single-user

In order to move quickly through testing and initial deployment, we began by implementing a simple web form that used straightforward encryption with common libraries to implement the architecture we had settled on. Namely, we created a PHP app that used libsodium's secret_box for public-key encryption.

Setup begins with the desktop application, which handles key generation and, later, decryption. A chosen person (or group) from the organization obtains the application (from our website, although this may be subject to change) and runs it on the computer where the long-term secret/private key will be stored. A single button press is all it takes to generate a

new cryptographic keypair, consisting of a public key file which can be uploaded to the web app, and a private key file which should be kept offline and backed up to a USB drive or other appropriate *private* backup location. We strongly caution against putting private key files on cloud storage.

From there, the organization's representative sent us the public key to upload to the web form (along with the form's questions to be asked, as we did not even have an online form editor at the time!). A public link could then be handed out to members, who submitted the form from their own devices. When submissions were received, HTTP POST data was serialized, encrypted with the uploaded public key, and appended to a submissions log for later bulk downloading.

Requests from other orgs

Around this time, we also began discussing our efforts on social media. This resulted in other organizations with similar needs and requirements reaching out to us to express interest in LockBox. After some internal discussion about the best way to address these requests, it was decided that LockBox would become a full-fledged Open Privacy application with support for multiple users/forms, a more polished user experience, and potentially even a hosted option for those who trust Open Privacy enough with their ciphertexts.

Multi-form, multi-user, hosted or self-hosted

While the organization's mutual aid fund was in full swing, we continued development on the prototype to make it a more professional-looking app that meets Open Privacy's standards.

The LockBox web app was upgraded from raw PHP to using the Twig template engine (the same one created for the Symfony PHP framework). User accounts and support for multiple custom forms was added, as was basic form editor functionality.

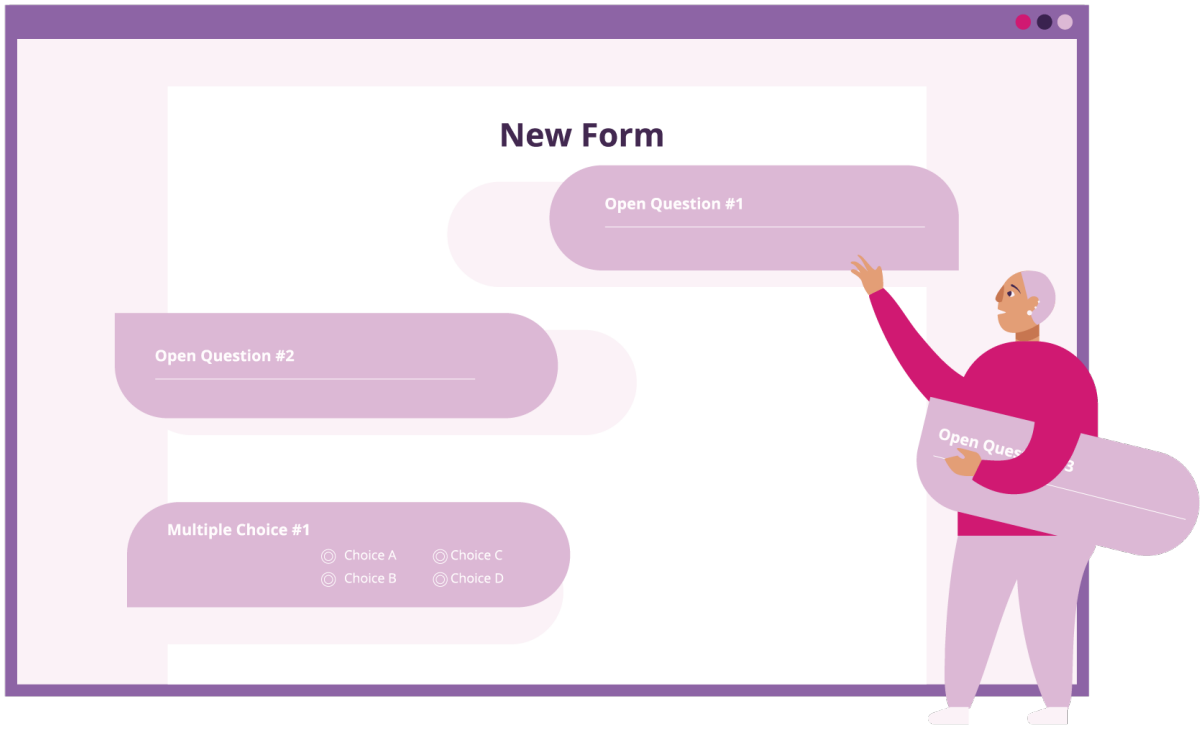
Also, thanks to some volunteer work from our board member (todo: permission/citation) who is working to set up LockBox for another community organization, we now support running the web app via popular containerization service Docker. :)

The LockBox desktop app was made using Open Privacy's Cwtch stack; native Go code with Qt bindings to handle a UI defined via QML. As mentioned above, we recently released Cwtch's widgets as a standalone QML library called Opaque, and this enabled us to rapidly

create an on-brand and professional-looking application with the flexibility of easily writing the encryption code using libsodium bindings for Go, a language the entire team is comfortable working with.

To help reduce friction in the setup process, we also added the ability for the desktop app to auto-upload public keys to the appropriate user and form on a specified (arbitrary) hosting server.

Fund administration



LockBox was given a minimalist, flexible design so that it can be more easily understood and thus adapted to an organization's specific needs. In what follows, we first explain the general intended operation of LockBox, followed by how the client organization chose to implement those operations.

LockBox Flow

The two cryptographic keys generated by the LockBox desktop app, one public and one private, can be managed in a variety of ways. In the "version 0" prototype of the app, the public key had to be manually uploaded to the webserver via a secure channel. In more recent versions, the user first logs in to the website, then retrieves a configuration token for the form they wish to configure. This token is entered into the app, and contains the information required to upload a fresh locally-generated key to the respective form on the webserver. Questions for the form are configured via the web interface similar to any other online form creator. This is all the setup that is required, and the public link provided by the website form manager can be shared with submitters.

Instructions (and now, our illustrated primer!) inform the user about the importance of managing the private key file. It should be kept offline somewhere it cannot be easily

accessed or copied by unauthorized users, but it should also be backed up, ideally to multiple places. Organizational characteristics should also be taken into account here. For example, should multiple people have a copy of the key in case one was unavailable for some reason? Should the key be kept somewhere that multiple people have to cooperate to access it? Should there be a log recording who accessed it and when?

Submitters do not have to install any software; the form is entirely online. The web app administration panel will generate a static public link that can be distributed via social media, printed on posters, etc. The form itself can be themed to match the organization's branding. Form submissions are encrypted in transit by HTTPS and then public-key encrypted upon receipt by the server before being stored to disk. (This creates a window of opportunity for a sophisticated hacker who has managed to compromise the server to replace or alter the software with something that exfiltrates the submitted data; see Future Plans for one potential way of mitigating this possibility.)

Encrypted submissions are stored in a log-like structure on the webserver with one submission per line/entry. This allows the web interface to inform the form admins of how many submissions have been received in total, in order to determine if new data needs to be downloaded.

If so, downloading encrypted submissions takes a single click and proffers a single cumulative file. By saving file this to the form's data folder that was autogenerated by the desktop app (or specifying its location manually), the app's "decrypt" function will become available and the submissions can be decrypted to a .csv (spreadsheet) file.

Organizational flow

Our client organization in this case chose to form a five member council that met weekly to review new applications and dispurse funds. Details and determinants of that decision-making process are beyond the purview of this case study.

Two members of their staff met virtually with the Open Privacy staff member developing the application to complete the setup process together. Keys were generated on a computer dedicated to the task, and backup copies created using USB keys. This proved prescient, as later, someone using the administration computer accidentally shift-deleted the private key and other configuration files. One of the backup USB keys was used to restore the private key and fund administration resumed unimpeded.

For the weekly review council meetings, a staff member would log in to the website shortly before the meeting and download a fresh copy of the form submissions. Decryption was reported to always proceed painlessly, and the resulting spreadsheet was handled in offline

Microsoft Excel. The need to meet remotely introduced by the pandemic at the time caused some awkwardness around distributing the submissions for review; they chose to err on the side of safety and have the staff member with the computer read relevant portions (e.g., non-contact information) of submissions to the rest of the council over secure video conference. Members met at a later date to review the files and confirm records against the data in person.

At the conclusion of the mutual aid fund's run, the web form was replaced with a thank-you message and the encrypted data was simply deleted once it was confirmed that all submissions had been downloaded and archived.

Analysis

Staff from both organizations reflected positively on the initial LockBox deployment experience. The version 0 prototype had such a small and easily-understandable source code footprint that Open Privacy staff were able to review it quickly, and take confidence in its security despite the atypically tight timeframe due to the relevant cryptographic bits being almost indistinguishable from libsodium example code. Other organizations were expressing interest in its potential to meet their own needs (and indeed, we have begun receiving contributions resulting from these new deployments getting underway!).

Development on our flagship project Cwtch (a free/open source metadata-resistant messaging platform) had resulted in a number of tools and libraries we hoped would make integrating Cwtch into other applications easier. However, we had yet to settle on exactly *what* those applications would initially be. While LockBox does not currently have Cwtch integration (it's on our "maybe one day" roadmap, see below), we were able to make use of Opaque, Cwtch's UI widget library that includes a configurable theme engine. We found we were able to use Opaque and our go-qt pipeline to create an initial functional prototype in under a day, and an application that matched Cwtch's somewhat unique aesthetic and we felt comfortable sharing with the client in a single weekend. Subsequent development proceeded slowly but steadily as a result of iterative interactions with and feedback from the client organization as they administered the fund.

Some complications arose around having to manually manage data files (a common pain point in cryptographic applications). The public key (and all other configuration files) were deleted by accident once; if our staff member hadn't been walking them through the setup process, they may have missed or ignored the instruction to make a back-up copy. As well, the review council initially struggled with how to meet online and share information about applicants without e.g. naively emailing the decrypted data around and undermining all the efforts to procure strong encryption and privacy. Recognizing the importance of communicating these subtleties of operation to future LockBox users, Open Privacy (with funding assistance from Privacy International) commissioned the creation of a narrated video, illustrated primer, and training materials for LockBox's setup and use. We are also considering plans to add data management and keyholding directly to the app, should our "Future Plans" (see below) for adding Cwtch peer-to-peer integration come to fruition.

The coincidence of a key loss happening on our very first trial run highlights the importance of helping users prepare for this eventuality. Open Privacy will be exploring integrating features such as threshold cryptography from our other technology portfolio for this, such as in our Shatter Secrets prototype which allows users to distribute a private key amongst an arbitrary number of remote parties, such that some minimum threshold of them must come together to combine their keyshares with each other before the key can be used to decrypt

any data.

As one might expect, administering such a mutual aid fund does not come without its share of external problems, and it is imperative that software developers in this space constantly be on the lookout that their efforts to reduce one harm do not perpetuate some other. In post-hoc interviews, staff said they did not believe LockBox or its technology platform caused, contributed to, or exacerbated any of the nontechnical issues they experienced. This was not an academic, market research or otherwise study and so we did not interview or interact with the client's members in any way, other than to exchange technical/debugging information in conversations relayed through a staff member.

The Future of LockBox

Open Privacy has already made significant progress developing the two LockBox applications. The web app supports potentially unlimited numbers of users and forms and has been released open source via our code repositories, making it suitable for self-hosting or easily hosting for others. The desktop companion app runs on Windows, Android, and Linux. Our in-house application stack already supports localisation and we just need to request translations via Lokalise.

In addition to requiring more polish and testing before being ready for full release, there are plenty of features we would like to add to LockBox given adequate time/funding, such as:

- Automatic download and decryption of submissions
- Potentially, in-app display of decrypted data (instead of exporting it to a spreadsheet)
- Forms with multiple pages
- Infrastructure for installing/updating the apps; possibly Windows code signing?
- More platform support: Mac and iOS (we have a new Flutter-based stack that theoretically should support this easily)
- Encrypted submission app: form submitters have the option of using an app to encrypt to the org's public key *before* submitting it!
- Server-side privilege separation improvements
- More field tests like the one described above, with more organizations
- Cwtch integration for form submission to enable peer-to-peer submission and/or key-sharing
- Source code security audit
- Opaque HTML edition (currently Opaque supports QML and Flutter)
- If we could move the form builder to the app, we could remove the web app's user system entirely... no more passwords!

Timeline

Open Privacy staff are currently focused on our flagship project, Cwtch, as it drives development of our libraries for speeding along other projects, and is near the core of our audience's support. We would welcome additional funding, and would be thrilled to be able to bring on capacity to turn any of the plans and dreams outlined above into reality, sooner. If you'd like to help speed this work along and help bring improved privacy and security to people around the world, please visit https://openprivacy.ca/donate or get in touch with us to discuss other options.