code{4}lib JOURNAL

Issue 29, 2015-07-15 ISSN 1940-5758

Exploring Information Security and Shared Encrypted Spaces in Libraries

Libraries are sensitive to the need to protect patron data, but may not take measures to protect the data of the library. However, in an increasingly collaborative online environment, the protection of data is a concern that merits attention. As a follow-up to a new patron privacy policy, the Oakland University William Beaumont Medical Library evaluated information security tools for use in day-to-day operations in an attempt to identify ways to protect private information in communication and shared storage, as well as a means to manage passwords in a collaborative team environment. This article provides an overview of encryption measures, outlines the Medical Library's evaluation of encryption tools, and reflects on the benefits and challenges in their adoption and use.

By Keith Engwall

Introduction

Privacy is a fundamental aspect of libraries. Information security has been a concern of libraries for over 75 years, particularly in terms of patron privacy, as evidenced in the very first Code of Ethics adopted by the American Library Association in 1939: "It is the librarian's obligation to treat as confidential any private information obtained through contact with library patrons" (ALA 1939). However, whereas information security in 1939 consisted mainly of locking paper records in a file cabinet, the challenges in today's digital world are far different.

In addition to protecting patron information, libraries must maintain other forms of sensitive information – account numbers, server passwords, etc. Keeping this information secure can be a challenge, particularly when it must be shared. Managing collaborative environments can be difficult in its own right, and most privacy technology is developed for use by individuals. On top of this, other more subtle issues may stand in the way of our efforts at information security. As our interactions through technology have increased, so has our acceptance of risk as it pertains to privacy.

In our personal lives, we conduct financial transactions, log our searches, map our comings and goings, and share the details of our day-to-day lives, exposing a trove of private information to the world without much consideration as to how it may or may not be protected. Meanwhile, at work, librarians seek opportunities to collaborate, and there are a plethora of free tools available to us for that purpose. We use tools like Twitter, AnyMeeting, Evernote, Trello, Google Docs, and others to seamlessly communicate, coordinate, and create across great geographical distances. In both cases, we allow ourselves to be fully reliant on policies and measures put in

place by others, often without sufficient understanding of the measures used to adequately evaluate their efficacy. In doing so, we prioritize the benefits we receive from these services over knowledge and/or control of the security protecting the information we put therein.

To be sure, we understand the difference between putting our own information at risk and that of our patrons, but an argument can be made that our relaxed attitudes towards our own information security may leave us less prepared when it comes to protecting the information of our patrons. Conversely, by developing better security habits for ourselves, we may improve our ability to protect the information of our patrons in our digital environment.

As librarians, we draw great benefit from collaborative efforts. Together, we are able to overcome limited time and resources to make better decisions, create stronger and more consistent policies, and conduct complex research on extremely limited budgets. However, due to limited budgets, we may not be able to avail ourselves of enterprise-level tools with built-in security, such as those used by corporations, hospitals, and other large institutions to protect the information of customers, patients, and corporate research and development.

Instead, we tend to rely on free or inexpensive software or services designed primarily for light, personal or casual use, such as Basecamp, Google Drive, Evernote, Trello, and others. Some of these services and/or software, such as Evernote, provide information about their security measures, while others provide little or no information. We must trust that our information is being protected and is not vulnerable to theft or disclosure. Fortunately, there are free and open-source tools available to allow us to take information security into our own hands. However, the burden of locating, implementing and using security software is on us.

In 2014, the Medical Library at Oakland University William Beaumont (OUWB) School of Medicine took part in a task force, alongside the broader University Library, to investigate patron privacy. The task force produced a joint patron privacy policy for the libraries, as detailed in a recent article in *The Journal of Academic Librarianship* (Hess 2014). As next steps in this process, the Medical Library was tasked with taking measures to identify gaps in information security and improving protections of patron information. As an extension of this process, we decided to consider other ways in which we might improve information security, not only for patrons, but for other areas, such as securing research data and internal information. By taking a broader, more comprehensive approach to information stewardship, we would develop better habits and be more successful in protecting not only our patrons' information, but that of the library and ourselves, as well.

The Medical Library decided to investigate how it might be able to provide a better information security infrastructure, not necessarily to dictate specifically how information is secured, but rather to provide tools to support information security. In some cases, available security measures do not fit into our current workflow, while others can be more readily implemented. We do not have any comprehensive solutions to the problem of information security, but we are taking steps to become more security conscious and will be evaluating our general information security practices alongside our efforts to protect patron privacy. Our investigations covered secure information, secure information storage, and password management.

Information Security Basics

To better understand how security software protects data, it helps to have a basic understanding of how data is protected. This section provides an overview of encryption and how it is used by security tools, such as those covered in this article.

Keys and Key Lengths

In encryption, an unencrypted, sometimes called plaintext, object, such as a document, file, or string of data, is scrambled using mathematical algorithms. The information within an object (characters, pixels, sequences of code, etc.) is represented digitally by a sequence of numbers. By passing these numbers through a complex mathematical algorithm along with a unique, random, very long number, they become scrambled, and thus, encrypted, as shown in Diagram 1. This unique, random, very long number is called a key.

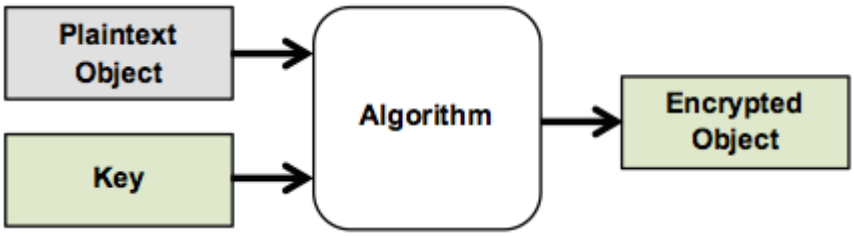

Diagram 1: An object is encrypted using an algorithm and a key.

The strength of the encryption is dependent on the length of the key. The length of a key is measured in bits, which use the digits 0 and 1. A key of length N bits has 2^N possible values (for example, a 1-bit key has 2 possible values: 0 or 1). A 128-bit key has 2^128, or 340,282,366,920,938,463,463,374,607,431,768,211,456 possible values. A 256-bit key has the square of that many values. The longer the key, the longer it would take someone to guess that key and decrypt the encrypted object.

Symmetric Encryption

Symmetric encryption uses the same key to encrypt and decrypt an object, as shown in Diagram 2.

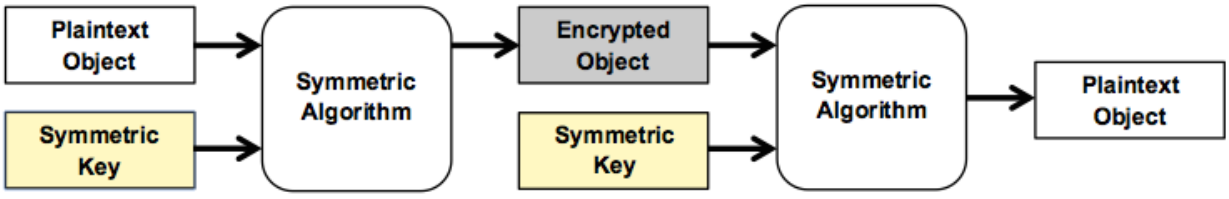

Diagram 2: The same symmetric key is used for encryption and decryption.

The Advanced Encryption Standard (AES) and Twofish are symmetric encryption algorithms. AES-128 or AES-256 are common implementations of this algorithm, using 128-bit or 256-bit keys, respectively. Twofish uses a similar naming convention (e.g. Twofish 256). Symmetric encryption is fast and very secure, and is what is typically used to encrypt files, text, and other objects. Each encrypted object is typically assigned a new key by the software handling the encryption, and may change each time the object is encrypted (Symmetric-key… 2015).

Asymmetric Encryption

Asymmetric encryption uses a pair of keys to handle encryption and decryption. It is often used in the transfer of data from one party to the other. Each party has a pair of keys. One of these keys is public and can be made available to anyone. The other is private and kept secret (Public-key cryptography 2015).

RSA Encryption

RSA (named for its authors: Ron Rivest, Adi Shamir, and Leonard Adleman) encryption uses one key to encrypt data and another key to decrypt it. Typically, the sender will encrypt an object using the recipient's public key, and the recipient will decrypt the object using their own private key. This is shown in Diagram 3.

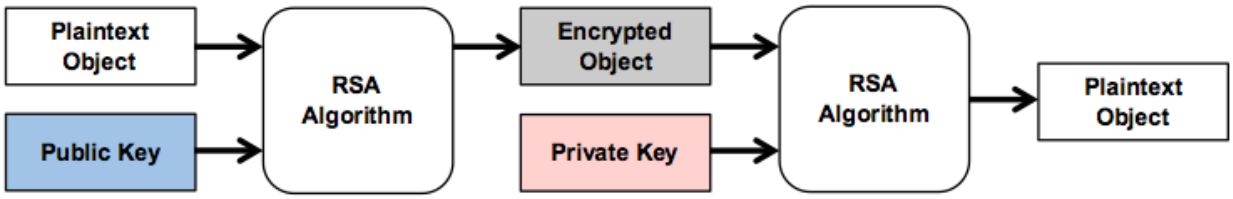

Diagram 3: In asymmetric encryption, different keys are used for encryption and decryption

Because asymmetric encryption is far slower than symmetric encryption, it is sometimes used to encrypt a symmetric key rather than the data itself, as shown in Diagram 4. The encrypted key is transferred and decrypted and then can be used to encrypt and decrypt the data, as shown in Diagram 1 above.

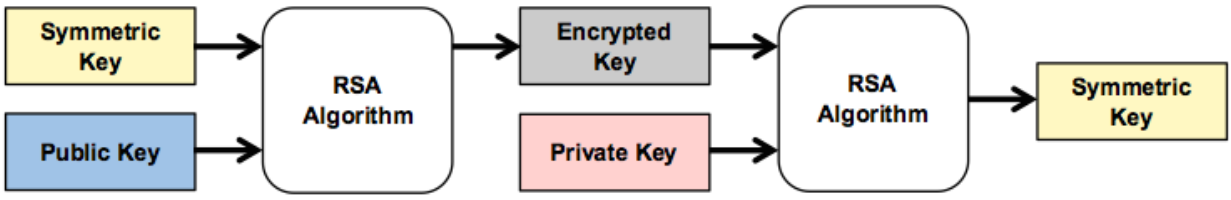

Diagram 4: RSA encryption can be used to encrypt symmetric keys, rather than data.

DH Encryption

DH (named for its authors: Whitfield Diffie and Martin Hellman) also uses asymmetric public and private keys, but combines each party's public key with the other's private key to create a secret shared key known only to the two parties, as shown in Diagram 5.

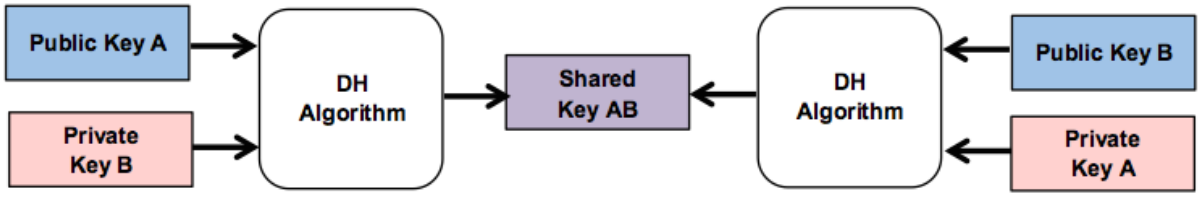

Diagram 5: DH encryption combines private and public keys into a shared secret key.

The shared key is then used as a symmetric key that both parties can use to encrypt and decrypt data transferred between them, as shown in Diagram 6.

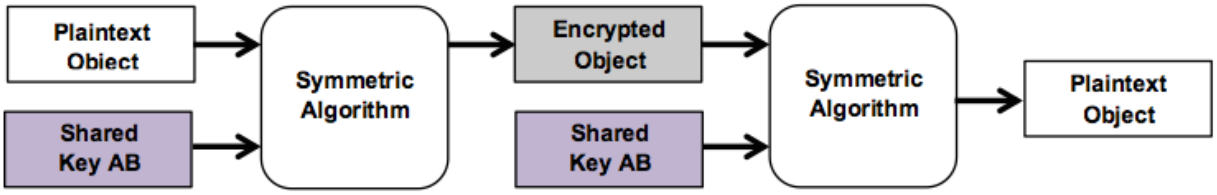

Diagram 6: The shared secret key is used for symmetric encryption.

Both RSA and DH make use of prime numbers, and since the number of prime numbers is relatively small in comparison to all numbers, asymmetric keys must be extremely long, anywhere from 1024 to 4096 bits long. In most cases, a new key pair is generated for each secure session.

There are many other encryption algorithms in use, but AES, RSA, and DH are very common.

Digital Certificates

Digital certificates are important to confirm the identity of a server with which a client is attempting to establish a secure connection. In these cases, the server's certificate is issued by a trusted Certification Authority, which can be checked during the initial connection to validate the certificate. A less common use of digital certificates is for users to create their own (self-issued) certificates for use with some secure communication protocols (Public-key certificate 2015).

TLS and SSL

Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols for secure communication (Transport… 2015). SSL has been in use since the mid 90's, but has been superseded by TLS. Both use certificates and asymmetric encryption to establish secure communication channels, but TLS contains a much larger set of encryption algorithms that the server and client may use for encrypted communication. When

you connect to a website with a browser, if the URL begins with https:// instead of http://, you are making a secure connection that will use either SSL or TLS, depending on the server and the browser (browser support of TLS has been ubiquitous for about a year).

Authentication, Hash and Salt

Just as digital certificates confirm the identity of servers, authentication is used to confirm the identity of a user on that server. Users create a password, which is passed through a hash algorithm and the hash is stored on a server. Unlike encryption, which is an algorithm intended to be reversible (decrypted), a hash is a one-way algorithm that can never be reversed. The output of this algorithm is a hash, which is stored on the server. Each time a person enters their password, it is passed through the same hash algorithm and produces the same hash. By comparing the hashed password with the hash on file, the server is able to confirm the identity of the user. This protects the user in the instance of a security breach. Even if an attacker gains access to the password hash file, they would still need the original passwords to gain access to user accounts. They would not be able to derive those passwords from the hashes, except through a "brute force" algorithm that guesses passwords, runs them through a hash algorithm, and compares the hash to the stolen hash table. A sufficiently strong password would take hundreds or even billions of years to guess.

By itself, a hash algorithm would generate the same hash for the same password chosen by multiple users. Repetition of hashes would tip off an attacker that those users have likely chosen weak and/or common passwords. To prevent this, extra data (usually random), called a salt, is added to the password before it is passed through the hash algorithm, thus producing different hashes. In order to replicate the hash during authentication, the salt must be stored on the server along with the username and password (Salt… 2015).

Multifactor Authentication

Another security measure that has started to show up as a feature in a variety of software and websites is Multifactor (or 2-Factor, 2-Step, etc.) Authentication. This is an additional layer of security that is used to confirm your identity.

This additional authentication step may be handled through a security question you set up ahead of time, entry of a code texted to your phone, a physical key such as an encoded USB device, or an application such as Google Authenticator. The first time you log in on a new computer or device, you are prompted to provide the second authentication step. You generally can choose at that time whether to register the new computer or device (such as when you purchase a new one), so that it is recognized on subsequent logins, removing the necessity of the additional authentication step. You can elect not to register the computer or device (such as when you are using a public computer), which will cause you to be prompted for the additional authentication step the next time you log in. The idea is that someone who had discovered your password would still be unable to authenticate without possession of a physical object, such as your phone or USB key.

How It Fits Together

So, how does everything fit together? Whether through a browser or through client software, initial connections to online services are typically encrypted using TLS or SSL, during which the validity of the server's certificate is likely to be confirmed, and private and public session keys are established and exchanged to create a secure connection. The user logs in with a username and a password. If the server uses salt, it uses the username to look up the salt for the hash. The password is hashed and compared to the hash on file. Once the user has been authenticated, access to the service is granted. All information transferred to and from the service will at least be encrypted during the transfer using the session keys established through TLS/SSL. Depending on the type of service being used (communication, social media, productivity, cloud storage, etc.) and the security policy of the service, information (files, images, text, etc.) stored within the service may or may not be encrypted on the server. If it is encrypted, it will likely be encrypted using symmetric cryptography, such as AES. This encryption will occur on the server and the service provider will be responsible for managing the keys for the encrypted data.

One major risk to storing sensitive information online is that an attacker who gains access to either the server or to the user's account may also be able to gain access to the unencrypted content. Last year, the discovery of the Heartbleed vulnerability, as described in a paper presented at the 2014 Conference on Internet Measurement (Durumeric 2014), highlighted how insecure this model can be. Through a bug in OpenSSL, an attacker would be able to silently and repeatedly gain access to the contents of server memory, around 64KB at a time, which may contain cryptography keys, usernames, and passwords. Thus, an attacker might be able to obtain keys to decrypt individual files (which would require gaining access to the server) or to a user's account. The latter scenario is particularly serious, since upon successful user authentication, the server would automatically decrypt any content requested by the attacker. Another security risk to consider is that if a service provider has full control over the content, it is able to decrypt information as it deems necessary, such as in response to pressure from government agencies. Although multifactor authentication would provide a measure of protection against vulnerabilities such as Heartbleed, it would not provide any protection against intentional disclosure.

Zero-knowledge Encryption

Zero-knowledge encryption is a term describing the process of encrypting information offline and storing the encrypted object on a server without also uploading the symmetric key. The server, therefore, has zero knowledge of the key and cannot decrypt the content. An attacker who gains access to the server or to the user's account would only be able to access the encrypted content, and since the key is never transferred to the server, it is not vulnerable to discovery through Heartbleed or other attacks. Software that encrypts information on the user's computer or device provides significant added security. Zero-knowledge encryption is not bullet-proof. Even though the security key is not stored on the server, a brute force hack of a password will generate the key necessary to gain access to the encrypted data. However, that is the only method of attack that will work. Other methods, such as tricking the service into letting an attacker change passwords or hacking the authentication process, may gain the attacker access to the service, but will not produce the key used by the local client to decrypt information contained within the user's account.

Medical Library Findings on Information Security

Secure Communication

One of the first areas we investigated was secure communication. We looked at various security measures we could apply to chat and email, but in the end did not pursue them for a variety of reasons. The following is a brief summary of our investigations.

OTR Chat

Chat can be encrypted using clients supporting the "Off the Record" (OTR) protocol. Programs such as Adium for Mac (v1.5.10) and Pidgin for Windows (v2.10.11) and Linux, support OTR, either natively or via plugin. OTR chat works using AES (128 bit) and DH (1536 bit) encryption. Each participant must generate a private key and a public key, called a fingerprint. OTR-compatible chat clients have options for generating these keys and handling the exchange through an authentication process. Once fingerprints are exchanged, participants can enter into a "private" (encrypted) chat session. Instructions for setting up OTR are available online (Hoffman 2014).

Although the Oakland University (OU) Library does use chat for reference and some internal communication, students and faculty in the medical school do not. For those who do wish to use chat to communicate, it is worth considering whether chat would be limited to internal communication or would be used with the public. Internal communication would likely occur within an institution's intranet and would likely be protected by its firewall. Also, policies could be put in place to omit sensitive information within chat. On the other hand, any reference chat with a patron could be considered, by its very nature, to be sensitive. Furthermore, libraries implementing reference chat would probably want to make it available to patrons outside the institution. In these situations, OTR-protected chat could be offered as an option. However, it would be up to the patron to install an OTR-compatible chat client, participate in the exchange of keys, and enter a private chat session. Currently, this is a fairly cumbersome process, but it is at least an option that libraries can offer. The Medical Library did attempt to use OTR chat to communicate internally with members of the University Library, but found that it was not a good fit, and since our patrons do not use chat either, we elected not to explore it further.

PGP-Encrypted Email

Along with walk-ins and phone calls, the Medical Library receives most of its interactions with patrons through email, either directly or to a reference email account. In our evaluation of patron privacy, we identified email as one of the primary ways in which personally identifiable patron information might be exposed. We investigated using Open Pretty Good Privacy (OpenPGP) encryption to encrypt email as a means to protect this information.

OpenPGP is not typically built into email clients, though several email clients are compatible with OpenPGP. In Windows, the Mozilla Thunderbird client (v38.0.1, www.mozilla.org/en-US/thunderbird/) can be set up to use OpenPGP through an extension. Mac Mail, built into the operating system, can be configured for OpenPGP by installing GNU Privacy Guard (GPG)

Suite (v2015.06, gpgtools.org). Browser plugins such as Mailvelope (v0.13.0, www.mailvelope.com) can be used on Chrome or Firefox to provide PGP support for web-based email. Although documentation is generally available from the creators of these products, the complexity of using OpenPGP is sufficient to merit additional instruction. Lifehacker has instructions for setting PGP up with Thunderbird (Trapani 2006), and a walkthrough for setting up OpenPGP on a Mac is available as well (Gangi 2014).

The OpenPGP standard supports several different kinds of symmetric and asymmetric keys, and these may vary depending on the program used. Similar to OTR-protected chat, using OpenPGP requires both parties wishing to encrypt an email exchange to install OpenPGP-compatible software and then create and exchange keys. Once that is done, someone writing an email could encrypt the content of a message for a specific recipient. This process is cumbersome to set up and use. Furthermore, a major drawback of OpenPGP-Encrypted email is that it only is able to encrypt content within the body of the email, not the email header. Unfortunately, the header contains precisely the kind of information (name, email address, subject) a library would wish to protect. This severely limits the protection OpenPGP can provide. Given these limitations and how cumbersome OpenPGP is to set up and use, we do not expect our patrons to embrace and use it. The Medical Library may look into it further, but for now we do not expect to implement OpenPGP-encrypted email as a security measure. Since our evaluation of OpenPGP, we have learned that Google uses TLS to secure email on its internal servers (Lidzborski 2014), though there is no guarantee that the email will remain encrypted when it leaves the Gmail servers. Outlook also has adopted TLS and other security measures (Sawers 2014), and we are hopeful that this is a trend that will be adopted more broadly.

That leaves the question of what to do about email. For now, the Medical Library is developing a set of practices to reduce the amount of patron email left lying around in our email accounts. First, we need to be able to identify email containing patron information. The University uses enterprise-level Google Mail, which allows you to apply labels to emails. By creating a Patron label and applying it to emails from patrons, we should be able to flag these emails. The next steps are to develop guidelines for retention and disposal of email, and decide how best to inform patrons as to good information security practices in their interactions with the library. We will review the institutional policy on data retention and determine whether it is sufficient.

We do not want to delete an email thread if we think that the patron might come back later with a follow up question. Our students conduct a Capstone research project that spans all four years of medical school, and they often need to revisit their literature searches from year to year. Faculty ask questions related to course development, and we may want to refer to research or recommendations in a previous year. Although we do not have a solution at this time, one thing we might do is apply labels to email threads that we believe to be complete and periodically review these to see which ones we feel comfortable deleting.

Another good security practice the Medical Library is trying to employ is to limit the amount of sensitive information contained either within the body of emails or as attachments. Instead, this information could be placed in a shared secure location and a link provided. Anyone intercepting the link would have to authenticate on the shared secure location in order to access the information.

Shared Secure Information Storage

The next facet of information security that the Medical Library investigated is secure information storage. While several options are available for encrypting information on a local computer, encryption of shared storage is more complicated.

To share files, the Medical Library uses shared folders on the University intranet and cloud storage services such as Google Drive (drive.google.com) and Dropbox (dropbox.com). The University maintains an internal encrypted storage service using Xythos software, which has since been purchased by Blackboard (www.blackboard.com/Platforms/Learn/Products/Blackboard-Learn/Blackboard-Xythos.aspx). Although the University recommends use of this service for secure online storage, the interface is clunky and storage is limited by default to 200MB. The University has an enterprise subscription to Google email, which includes Google Drive, which recently increased its quota from 30GB to unlimited storage, and we make heavy use of the collaborative features inherent in Google Documents, Forms, etc. Although the University discourages the use of Dropbox, some Medical Library staff use it for collaborative projects managed by others outside of our institution.

The University encrypts its centrally-managed shared file servers, which are accessed through network shares. Remote access to these folders requires a Virtual Private Network (VPN) connection. When working with groups outside of the institution, it is sometimes necessary to use cloud storage services such as Google Drive and Dropbox, since individuals not affiliated with OU do not have access to the University intranet. Although these services may encrypt data, that encryption is performed by the vendor and the keys are stored on their servers. Given that users have no knowledge or control over the encryption process, there is no way to be confident that data stored on these services is truly secure. The Medical Library investigated a variety of zero knowledge encryption options for encrypting cloud data, including software that encrypts data locally for existing cloud storage, as well as stand-alone cloud storage services that provide a way to encrypt data before syncing it to the server.

It should be noted that data stored in the cloud, even encrypted data, may not satisfy security requirements of your Institutional Review Board (IRB) for personally identifiable data in research. Furthermore, it may not meet requirements for protecting patron, patient or student data. There are still situations, however, when you may wish to secure information that you share with colleagues, and the following solutions should suffice for this purpose.

The Medical Library investigated both encrypted cloud storage services and encryption software to use on Windows and Mac with either Dropbox or Google Drive. For testing, we used two computers, a 2011 iMac running OS X 10.10.3 Yosemite and a 2013 Macbook Air running Windows 7 Enterprise (Service Pack 1). All of the solutions we tested use AES-256 encryption and RSA encryption. In addition to working from our own experience with encryption software, we reviewed online recommendations for secure online storage, including an article from Lifehacker (Henry 2013), and selected the following solutions to evaluate.

Encryption Software

Viivo by PKWARE, Inc. (v2.6) – viivo.com

Viivo is commercial software with a free version whose terms of service do not restrict how it is used (personal vs. business). Additional features and support are available with paid subscription versions. Viivo uses AES-256 and RSA (2048 bit) encryption and works by syncing files between a decrypted folder and an encrypted folder. Specifying a cloud service will place the encrypted folder within the local files for a cloud service such as Google Drive or Dropbox. In testing, we found Viivo to be extremely clunky, and the software does not seem to be fully developed. We had great difficulty getting the Mac version to connect to Google Drive and were not successful in getting multiple users to access and decrypt content on Google Drive (although it is unclear whether the issue was with Google Drive or Viivo, we did not have difficulty connecting to Google Drive using other software). We had better luck with Dropbox, and were able to set up an encrypted share between our Mac and Windows computer, but the process for setting up the share was too cumbersome to expect others to tolerate.

Cloudfogger by Cloudfogger GmbH (v1.4 for Windows, v1.0 for Mac) – cloudfogger.com

Cloudfogger is free commercial software, primarily for Windows. Cloudfogger uses AES-256 and RSA (unspecified) encryption. The Windows version works by monitoring a folder on your computer and automatically encrypting every file and subfolder within. So long as Cloudfogger is running in the background, it will automatically decrypt a file when you access it. By monitoring a folder within the local files for a cloud service, those files are encrypted and only accessible while Cloudfogger is running. If you share a folder with someone through your cloud service, you can add them to a share list in Cloudfogger, and when they log into their Cloudfogger account, they will be granted access to the folder. Unfortunately, the Mac version has none of these features and only allows you to manually decrypt and encrypt files.

Boxcryptor by Secomba GmbH (v2.1) – boxcryptor.com

Boxcryptor is free for personal use on a single device with a single cloud storage provider. Paid subscriptions are available for personal use on unlimited devices with unlimited cloud storage providers and for unlimited business use. Boxcryptor uses AES-256 and RSA (4096 bit) encryption, and works by mounting a virtual drive on your computer (through a drive letter in Windows and through a mounted volume on Mac) that provides decrypted access to an encrypted folder. Files added to the virtual drive are automatically encrypted and can only be viewed by accessing them from the encrypted drive. In Boxcryptor's settings, you select a cloud service (the paid version lets you select more than one), and the local folder of that service is shown in the virtual drive. Any file or folder within can be encrypted. Unfortunately sharing is only available with the paid version, which is prohibitively expensive for our purposes.

Encrypted Cloud Services

SpiderOak by SpiderOak, Inc. (v5.2) – spideroak.com

SpiderOak provides 2GB of online storage for free, with additional storage and features available at various levels of paid subscriptions. Terms of Service indicate no restrictions on type of use

(personal vs. business) for the free version. Client software creates a local unencrypted folder on your computer. Files placed within the folder are synced up to the cloud service after being encrypted locally. SpiderOak uses AES-256 and RSA (3072 bit) encryption. Unfortunately, in testing, we discovered that the only sharing available in the free version is to provide read-only access to others through a URL, which would not be sufficient for collaboration.

Tresorit by Tresorit AG (v2.0) – tresorit.com

The free Tresorit account provides 3GB of storage and limits some features, with additional features and storage available for a paid subscription. Terms of Service indicate no restrictions on the type of use (personal vs. business) for the free account. Tresorit uses AES-256. The Tresorit client lets you access and manage containers called Tresors (German for "vault"), each of which may be configured to sync to a folder on your local computer. If you do not wish to keep a local decrypted copy of the files on your computer, you can turn sync off and access the files through the client. Each Tresor can be shared with other Tresorit users. The free account lets you have three Tresors, each of which can be shared by up to 10 users. The Tresorit client has a clean and simple interface, is easy to set up on both Mac and Windows, can be configured to use multifactor authentication (explained further in the section on password management) for extra security, and can be configured to automatically run and log in on startup, allowing you to interact with files in a synced Tresor folder without touching the client.

Based on our investigations, we found Tresorit to be the most promising free option for setting up an encrypted share for a small group. Sharing with larger groups would only require the owner of the share to pay for a subscription, allowing other users to access the share for free.

Password Management

The Medical Library was also interested in using a shared password management system to allow the library and smaller groups therein to manage passwords for services, such as the reference email account, and other resources, such as web servers and database users. As it happened, the University Library had already investigated password managers for its computer support department, including KeePass, DashLane and LastPass. After some discussion, it was decided to share a group purchase of a password manager.

KeePass Professional Edition (v2.29) – keepass.info

KeePass Professional Edition is free, open source software that runs under Windows. It supports AES and Twofish symmetric key encryption. KeePass works by creating an encrypted database which can be unlocked using a master password and/or a key file. Records containing information such as the name of the program or website, login, password, website URL, etc. are stored in the database. KeePass records must be created manually, but KeePass does have features to autofill logins and passwords if you initiate connection to a web site from the program (which opens the site in your default browser) and there are third party plugins to provide additional browser integration. A KeePass database can be shared by storing the password database in a shared location. Multi-user support is provided through plugins for the program. Since KeePass is locally installed software, no user registration is required, but multiple users

must share a single password to access the database. KeePass uses the .NET environment, which is built into Windows (Vista and higher). It does run on Mac computers, using Mono (v2.6 and higher), an open source .NET framework. However, according to the KeePass website, the official site supporting the Mac version of KeePass is no longer available. An open source project, KeePassX (alpha v0.4.3), based on KeePass 1.x, is available for the Mac. The developer describes it as a hobby project, and it hasn't been updated in four years, but the underlying code is relatively stable.

Dashlane by Dashlane, Inc. (v3.5 for Mac, 3.2.5 for Windows) – dashlane.com

Dashlane is a password management service that provides a free account for personal or internal business use. It runs on Windows, Mac, iOS and Android. Dashlane is free when used on a single computer or device. For a subscription, Dashlane Premium provides syncing between computers/devices, unlimited sharing of passwords, and other features. Dashlane integrates well with a variety of browsers. It uses AES (256 bit) to encrypt its database, and even if you subscribe to the Premium version, you can choose to restrict your database to your local computer (instead of uploading the locally encrypted database to its servers). Dashlane uses an auto-create feature to add records to its database when you log into a website, as well as an auto-fill feature to enter login and password data into a website from an existing record. Individual password records may be shared with other users. Dashlane requires users to register in order to create an account, but authentication is handled locally (no internet connection is required to open the password database). It supports multifactor authentication.

LastPass by LastPass (v3.1.95) – lastpass.com

Similar to Dashlane, LastPass is a password management service that runs out of your browser through a toolbar icon. The free account allows you to install LastPass on multiple computers and provides web access to your password database. Different subscription levels provide additional features such as use of mobile apps (Premium subscription) and sharing features (Enterprise subscription). It uses AES (256 bit) to locally encrypt the password database before uploading it to its servers. LastPass does not provide a local-only option for its database. It provides similar auto-create and auto-fill features as Dashlane. It also supports multifactor authentication. LastPass Enterprise provides a robust set of features. Individual records can be shared, or shared folders can be created to organize records and automatically provide access to users with whom a folder is shared. LastPass users who have a personal account as well as an Enterprise account can link the accounts so that when they log into their Enterprise account, they will be able to access their personal database as well. LastPass can audit your password database and alert you to bad security practices (weak passwords, using the same password for multiple accounts, etc.).

All of the above password managers were given consideration. At least one person on staff already used each one. Given the limited sharing capability of KeePass, as well as the uncertainty of Mac development, KeePass would not work well in the Medical Library's Mac-heavy environment. KeePassX was excluded from consideration due to its lack of development. Dashlane was more promising, but LastPass was ultimately chosen because of its more robust

sharing capabilities and the proactive way in which LastPass handled the Heartbleed vulnerability (Kovach 2014).

Conclusion

After evaluating the above security measures, we have had success using LastPass as a shared password manager, but have not found ways to fit other measures into our workflow. We ended up not using chat at all and were disappointed in the email encryption options available because they are cumbersome and do not protect the identities of the sender and recipients. We are encouraged that email encryption is starting to take hold among major providers, but it is unclear how quickly this will be more broadly adopted or how transport security might apply to stored emails. For file storage, we continue to primarily use institutional file servers to store internal, sensitive information. We are hoping to pilot use of Tresorit with an external group if we are able to find willing participants.

Still, we are pleased with the results thus far. About half of the staff in the Medical Library have begun to use LastPass on a daily basis. It integrates very well with the browser, so much so that one of our staff didn't even realize they were using it. We have two shared password folders, one for the Medical Library, and one shared with the University Library's technology team. The technology team has made the heaviest use of the shared folders, mainly storing server passwords. The Medical Library has begun to populate its shared folder with passwords for shared services (such as our shared reference email account). Access to passwords and folders can be configured to specific individuals, so only library staff that are authorized to have access to library resources are able to access the usernames and passwords for these resources. This, in itself, is a great benefit to managing sensitive library systems. Previously, passwords would need to be exchanged verbally or written down, each of which has some degree of risk.

At the time of the writing of this article, the Medical Library received a security alert from LastPass (Siegrist 2015), notifying us that an attack was detected on their system and user password hash files and salts were compromised. The service immediately implemented a security measure, requiring any logins from non-trusted (new) computers to undergo an email verification step, unless the account already had other multi-factor authentication in place. The alert reiterated the purpose of strong passwords in the protection against such an attack, and advised users to change their account password in any case. Our site administrator was able to require us to change our passwords, even though some of our passwords were strong enough that they would be at a very low risk from brute force guessing. Although we were concerned about the breach, we felt that the response was timely and sufficient to protect our data.

The steps we have taken thus far are small ones, and do not have a significant impact on the protection of patron information itself, for which the library continues to rely primarily on security built into library systems such as the Integrated Library System and Interlibrary Loan System. Still, they move us towards building a more comprehensive approach to information security. One of the next steps recommended by the Libraries' Patron Privacy Policy is to develop security procedures for the identification, management, and retirement of patron information, as well as a means to monitor how well the library is implementing these procedures. While use of additional security measures is outside the scope of the patron privacy

policy, the Medical Library will look into how we might support its use through training, creation of guides, etc.

One thing has become clear through this process. It is very easy to take digital information for granted and fail to give due consideration to its vulnerability. The steps we have taken thus far are modest, at best, but are also important in that they encourage us to take a proactive approach to thinking about information security, and, we hope, prompt us to take preemptive action. The first step is the most critical in protecting the sensitive information of our patrons, our institution, and ourselves.

References

American Library Association (ALA) 1939. Code of ethics for librarians. American Library Association [Internet]. [cited 2015 May 12]. Available from: http://www.ala.org/advocacy/proethics/history/index5

Durumeric Z, et. al. The matter of heartbleed. Proceedings of the 2014 Conference on Internet Measurement. 2014 [cited 2015 Jun 15]: 475-488. Available from: http://dl.acm.org/citation.cfm?id=2663755

Gangi J. The best PGP tutorial for Mac OS X, ever. Jerzy's Notes [Internet]. [updated 2014 Mar 12]. [cited 2015 Jun 15]. Available from: http://notes.jerzygangi.com/the-best-pgp-tutorial-for-mac-os-x-ever/

Henry A. The best cloud storage services that protect your privacy. Lifehacker [Internet]. [updated 2013 Jul 10]. [cited 2015 Jun 15]. Available from: http://lifehacker.com/the-best-cloud-storage-services-that-protect-your-priva-729639300

Hess AN, LaPorte-Fiori R, Engwall K. 2014. Preserving patron privacy in the 21st century academic library. The Journal of Academic Librarianship [Internet]. [cited 2015 May 12]; 41(1). Available from: http://hdl.handle.net/10323/3345

Hoffman C. 2014. How (and why) to use OTR for private instant messaging. HowToGeek [Internet]. [cited 2015 May 12]. Available from: http://www.howtogeek.com/190811/how-and-why-to-use-otr-for-private-instant-messaging/

Kovach S. Find out instantly if a site has been infected by 'Heartbleed'. Business Insider [Internet]. [Upated 2014 Apr 9]. [cited 2015 May 12]. Available from: http://www.businessinsider.com/lastpass-heartbleed-checker-2014-4

Lidzborski N. Staying at the forefront of email security and reliability: HTTPS-only and 99.978% availability. Official Gmail Blog [Internet]. [updated 2014 Mar 20]. [cited 2015 Jun 15]. Available from: http://gmailblog.blogspot.com/2014/03/staying-at-forefront-of-email-security.html

Public-key certificate. Wikipedia [Internet]. [updated 2015 May 16]. [cited 2015 Jun 15]. Available from: https://en.wikipedia.org/wiki/Public_key_certificate

Public-key cryptography. Wikipedia [Internet]. [updated 2015 Jun 4]. [cited 2015 Jun 15]. Available from: https://en.wikipedia.org/wiki/Public-key_cryptography

Salt (cryptography). Wikipedia [Internet]. [updated 2015 Apr 29]. [cited 2015 Jun 15]. Available from: https://en.wikipedia.org/wiki/Salt_%28cryptography%29

Sawers P. Microsoft boosts Outlook.com email encryption and opens its first transparency center. TheNextWeb News [Internet]. [updated 2014 Jul 1]. [cited 2015 Jun 15]. Available from: http://thenextweb.com/microsoft/2014/07/01/microsoft-brings-tls-encryption-outlook-com-opens-first-transparency-center/

Siegrist J. LastPass security notice. LastPass Blog [Internet]. [updated 2015 Jun 15]. [cited 2015 Jun 17]. Available from: https://blog.lastpass.com/2015/06/lastpass-security-notice.html/

Symmetric-key algorithm. Wikipedia [Internet]. [updated 2015 Jun 15]. [cited 2015 Jun 15]. Available from: https://en.wikipedia.org/wiki/Symmetric-key_algorithm

Transport layer security. Wikipedia [Internet]. [updated 2015 Jun 15]. [cited 2015 Jun 15]. Available from: https://en.wikipedia.org/wiki/Transport_Layer_Security

Trapani G. How to encrypt your email. Lifehacker [Internet]. [updated 2006 Jun 15]. [cited 2015 May 12]. Available from: http://lifehacker.com/180878/how-to-encrypt-your-email

About the Author

Keith Engwall, MS LIS, AHIP is an Assistant Professor at the
Oakland University William Beaumont School of Medicine, in Rochester, MI. He serves as the Web and Emerging Technologies Librarian at the OUWB Medical Library. He can be reached at engwall@oakland.edu