Introducing the cipher mail transport protocol (CMTP)

Jonathan Moroney
University of Hawaii at Manoa
jmoroney@hawaii.edu

January 21, 2016

Contents

Abstract

Email is great, but it sucks. I want to take the sucking out of email and I propose to do so by creating a new mail transport protocol.

1 Introduction

Email is an integral part of modern communications and daily life. It's used for nearly all formal communications and it's a broken system. Email has evolved from the ancient times to provide an asynchronous messaging system where users are tied to servers which facilitate mail and servers are tied to each other to ensure that they can facilitate mail. This relationship between the user and their server is something that allows mail to work, but at the same time is something that can be revised. For example; a generic user will trust their server to (attempt to) deliver any mail they send to it, they will trust it to receive mail on their behalf and to occasionally forward their mail to another server. To the user a mail server is a diligent butler. However, as has become evidently clear in the past few years, users are trusting their infrastructure far more than is reasonable.

The infrastructure in question dates back to 1982 when SMTP was first defined and stretches to the present day with ESMTP, POP, IMAP, OpenPGP, and webmail[6]. This infrastructure includes many disparate servers across the globe talking the same language and making use of other internet infrastructure such as the DNS system. I love this infrastructure and think there is a great merit in allowing any idiot out there to become a node, but this is also where the trust issues stem from as some idiot might not be so idiotic and might be quite hostile. As such, it is time for a new protocol that acknowledges the hostility of the modern network.

1.1 Modern Email

For the purposes of this paper "modern email" will refer to SMTP used in conjunction with IMAP or a webmail client. POP, while historically important, is not considered to be modern. Modern email is a system with which users can send and receive email from many devices/programs, can change programs/devices without administrative approval, and can deal with email in an asynchronous fashion. It's important to understand that email is a generic term for a number of different systems and that by modern email; I mean none of them specifically, but all of them generally. That is, all of their "good" parts generally. For instance, SMTP originally required every user to have an SMTP server of their own and this I qualify as bad. Conversely, gmail allows a user to travel the globe and to have full access to their communications from anywhere so long as they know some login information; this I qualify as good. It's unfortunate that the good quality of email that I bring up can be exploited if you are: in the wrong country, are the wrong person, cross the wrong IT guy, are at the wrong coffee shop, etc...

1.2 Motivation

This section may be of a very different tone from the others, but I wanted to include it so that my impetus is clear. Encryption has been an interest of mine for about as long as I've been aware of it, and secure messaging the most interesting application. I used to think that encryption worked quite well until I actually had to use it in *"the real world"*

```
A few years ago, I had the opportunity to be an intern at a large
    tech trade show (Interop) and it was a great thing to have done.
     However, in order to get reimbursed for hotel, flight, and the
    like I had to get in contact with a woman in New York. She
    needed to fill out some forms with information that I'd rather
    not be in plaintext as it traveled a quarter of the way around
    the globe. So, I encrypted it in a password protected zip and
    sent it on its way. What followed was a week of this woman being
     pissed at me for making her life that little bit harder and
    some awkward social situations that I had to navigate while at
    the show.
```

It's probably not a unique story and it's easy to just place blame for the situation and move on. But, it's the triviality of it that stuck with me and it's the infrastructure that was at fault. Couple that with being in grad school and needing a thesis project and we arrive at this document.

2 Related work

Securing communication has been a topic of active research for about as long as communications have been written. Similarly, securing email has been a topic of active research for about as long as email has been around. The most prominent email security systems are the OpenPGP and the S/MIME standards. To the best of my knowledge both standards provide good security, but neither has been able to make security a standard feature of email. By contrast the Darkmail Alliance has introduced the Dark Mail Transfer protocol (DMTP) and the Dark Mail Access Protocol (DMAP) which aims to partially anonymize conversations, create a trust chain for key rotation and more[5]

2.1 OpenPGP

The open pretty good privacy (OpenPGP) standard serves as the primary inspiration for the cipher mail transport protocol and that's because the author of this document thinks OpenPGP is a brilliant system. OpenPGP works by having users exchange independently created public keys which are then used to encrypt mail and pass it in the normal SMTP way. OpenPGP is a decentralized system that doesn't require users to trust an arbitrary authority, but does allow for the development of a web of trust through

shared contacts[4]. The fundamental problem with OpenPGP is simply that it has never reached universal adoption and so users have always had to be aware of who they are talking to (on a technical level). As a result of non-universal adoption are a number of usability problems, including, but not limited to, requiring users to manage their own keys, requiring users to manage their own key exchanges, support for broken ciphers, and a lack of clean mail client integration.

2.2 S/MIME

If OpenPGP is the bazaar then the Secure/Multipurpose Internet Mail Extensions (S/MIME) is the corresponding cathedral with a top down command and control key exchange system. In an S/MIME environment users define a trusted key source and then software will sort out all the key exchange and usage[7]. Problems with S/MIME come from the same lack of universality as with OpenPGP and include cross client compatibility, inter-domain key exchange, and communication with non-S/MIME users.

2.3 Darkmail

The Dark Internet Mail Environment (DIME or Darkmail) is a system developed by Ladar Levison in response to the Lavabit incident[2]. Darkmail aims to prevent mail operators from being in the position where they become informants on their users. Ladar has designed Darkmail to be anonymous at the domain level in addition to the encrypted content of the messages. Additionally the idea of a 'signet' is introduced as a sort of super key which includes an amount of plaintext information (Name, Address, Phone Number, Motto, and many more) along with a cryptographic key. The idea of the signet is that the addition of the plaintext information make keys easier for a human to accept. The problem with Darkmail is that it aims to solve every security problem that email faces and as such Darkmail has yet to make a meaningful impact on the world. In fact as of January 21, 2016 the Darkmail website (`http://www.darkmail.info`) is down.

3 Security Model

The cipher mail transport protocol (CMTP) aims to provide end to end encryption and message security regardless of network security. However, unlike many crypto systems CMTP does not try to provide identity verification and this is done to maintain the usage model of email. Why is this? Well, I believe that there's a cost associated with providing identity verification and that cost translates into a user interface that becomes unpalatable for the lay user. The lay user wants an email infrastructure that is reliable, works for new contacts, has that certain sense of officiality and is

private. What are the attack vectors on modern email? Well, email at its' most basic level is a file transfer system. One user creates a file and passes it to another after annotation, editing, and perhaps conversion. We can't protect anything at either end as either user is free to share the file as they see fit, but everything between the two users can be secured.

3.1 Threat model

It would be easy to say that for a given message the attackers are simply everyone who is not either the sender or the receiver, but this ignores the fact that Alice and Bob are idiots and can't be trusted to properly manage their keys or to reason about security. So from the point of view of protocol design we should aim to minimize what we need from our Alices and Bobs to use our system and we should treat user error as a weakness in the security system. The crypto system being used for encryption mode one is assumed to be secure[3] which leaves operational security left to be considered in this protocol. The primary known weakness in CMTP comes from the key requests which can be intercepted by an attacker who has become a man in the middle of two CMTP servers. The attacker is able return any key she desires to the requesting party and thus can own the communication channel. For now this is not a problem I propose to solve as

- Solutions I can think of break the modern email usage model
- This attack is costly as the attacker would need to pass all mail in a timely manner to remain unknown.

Moving on to key storage in general we need to maintain key pairs for users over a long period of time. One weakness of GPG is that losing a key is very easy to do and by design there is no key retrieval mechanism. This leads to scenarios in which users can communicate with each other, but must be willing to accept new keys for contacts they already know. This is a problem as it either makes the man in the middle a trivial attack or leads to socially awkward situations. To avoid this problem CMTP tasks the CMTP server with being a key server for both public and private keys. But, a user should never share their private key since everyone is considered to be untrustworthy with respect to a private key.

3.2 Xzibit key storage

"Yo dawg I heard you like encryption, so I put some crypto in your crypto so you can encrypt while you encrypt".

Put more formally; we can task an untrustworthy server to hold users private keys if the private keys are themselves encrypted. This has two benefits to the user:

1. It lowers the cognitive demand on the user. They only need to know a single password.

2. It maintains the usage model of modern email. A user can retrieve their keypair from any machine, anywhere in the world.

So, during account creation a CMTP client randomly generates a public/private keypair using only the best quality, free range, organic random numbers and then asks a user for a password. The user's password is used to encrypt the private key. At this point the CMTP client can send both the public key and the encrypted private key to the CMTP server for storage without fear of leaking secrets. It's worth noting that for political reasons users should not trust their servers any more than they have to as service providers may occasionally be their enemies[1].

3.3 Usable security

As mentioned above trusting any key that gets returned after a key request is insecure. When accepting keys it's possible that an attacker could intercept the key request and return a key of their choice. This key interception could lead to a man in the middling of a conversation and it is clearly a security concern. However the usage model of modern email allows two users to have a conversation without prior knowledge of each other and without a shared secret. I aim to maintain the usage model of modern email and thus an insecure choice has been made in the design of this protocol. In general when faced with a question of usability v.s. security this protocol prefers usability.

4 Cipher mail transport protocol

The cipher mail transport protocol (hence forth CMTP) has two goals

First
: To maintain the usage model of modern email

Second
: To maximize security while not violating the first goal

4.1 High level design

At the high level CMTP works by ensuring that all users have public/private key pairs and by tasking the infrastructure with transporting public keys as well as messages. Servers have their own keys which allows them to trust and keep track of each other. Users generate their keys client side, encrypt the private key with a symmetric cipher and task the server with holding

the keypair. In a webmail scenario this should be done in javascript or some other client side language so that the server never sees an unencrypted private key. Having the server hold both keys allows a user the freedom of modern email where they can have multiple devices, add a new device without talking to a server administrator, and can expect to get any public key for any user at any time.

4.2 CMTP interface

Inspired by SMTP a CMTP connection is driven by a text interface which readies client and server for a binary message transfer. Unlike SMTP, CMTP has no secondary commands. That is, each command is a stand alone operation and no state need to be maintained between commands. Commands are capitalized ascii strings terminated with the new line character (<CRLF>) and after the OHAI command there is no ordering to the commands; each can be thought of as an atomic operation. All commands should be acknowledged before the client sends anything more. This text interface is the set of CMTP commands which are

OHAI
: Used to begin a connection. Once received the the server assumes it is talking to a CMTP speaking entity.

MAIL
: Used to indicated that the next bytes received are a self describing message.

HELP
: Used if you haven't read this document and/or would like to poke the server.

NOOP
: Used for testing.

OBAI
: Used to end a connection.

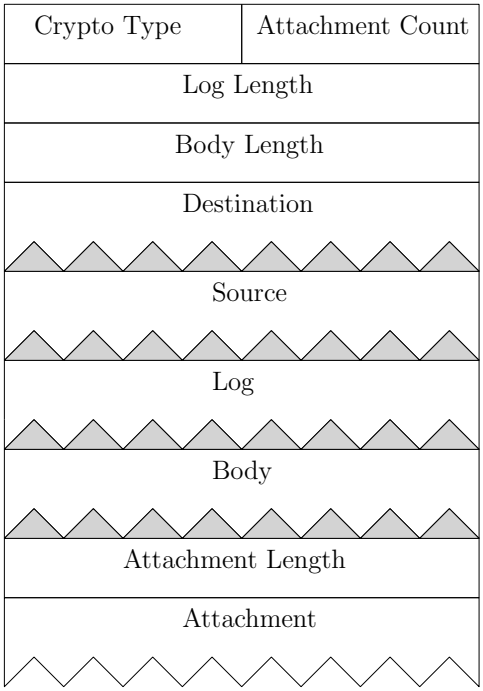
KEYREQUEST
: Used to get a servers public key. After a client issues the KEYREQUEST command the server will reply with it's crypto type and public key.

In both the MAIL and KEYREQUEST cases the following data will be a CMTP message which is a self describing data structure.

4.3 Message structure

The structure of the CMTP message can be broken down into two large types. The plaintext "envelope information" and the cipherbytes which constitute the body and the attachments. The log structure is plaintext and is used to track forwarding, acceptance, and rejection by CMTP servers. The fields that begin the CMTP message are the fixed length fields and their position in the file structure is for programming convenience. The variable

Figure 1: CMTP message structure. *Subject to change*



length fields described below are considered to be UTF-8 encoded strings that are null terminated. That is; the first byte [00000000] defines the end of that field. The only field for which this is not true is the attachment field(s) which are to be treated as binary blobs. Since the number of attachments is variable each attachment field is preceded by an attachment length field. The fields are used as follows

Crypto type - 4 bytes

Should be interpreted as an integer type and is used as a lookup for the crypto scheme used for the message. 0 is defined as "encryption mode zero" which means that the message body and attachment body are plaintext (UTF-8 and binary blob respectively). encryption mode zero is not implemented in the reference code, but is defined here for future use. 1 is defined as "encryption mode one" (there's a pattern here)

and is whatever libsodium provides via the secret boxes api *need to define this better*.

Number of attachments - 4 bytes
Should be interpreted as an integer and denotes how mant attachment blocks follow the message body.

Log length - 8 bytes
Should be interpreted as an integer and is the count of bytes in the log field.

Body length - 8 bytes
Should be interpreted as an integer and is the count of bytes in the body field.

Attachment length - 8 bytes
Should be interpreted as an integer and is the count of bytes in the following attachment field.

All integer types are big endian so that they agree with network order. Additionally the attachments have an internal structure which is

- Filename, the first 255 bytes in UTF-8 encoding
- Attachment body, the remaining attachment length - 255 bytes

4.4 Key requests

The key request is a special case of the general Cipher Mail Transport Protocol (CMTP) message. In any given key request there are two parties; one who has a particular key and one who desires this particular key; the two parties will be two CMTP servers requesting keys on behalf of one or more of their users. The CMTP message format should be the same as a standard mail message with the destination account being null (that is one byte with the value zero) for whatever server the request is for. Further the body of the message should be a null terminated list of accounts for which public keys are being requested.

Key requests need to be replied to and the format of the reply is as follows
KEY_DELIVERY
account1@example.com\0CRYPTO_TYPE
KEY
account2@example.com\0CRYPTO_TYPE
KEY
The crypto type defines how long a key is so even if the key allows for a newline character so, it's clear what part of the message is key and what part is not.

5 Reference Implementation

Reference code is provided at `http://www.not.online` and is in two parts.

- cmtpd - The reference ctmp mail server
- cmtp_client - The reference cmtp mail client

6 Future work

6.1 Inside scope

Work has already begun on building a free (as in gnu) CMTP server and client and basic command line tools to send and receive mail will be completed. The initial client mail retrieval will be fairly basic as most of my work is being put into the server.

6.2 Outside scope

1. Add SMTP command support to cmtpd
2. Create an IMAP like protocol for CMTP messages
3. Solve the key request insecurity
4. Implement "encryption mode zero".

References

[1] China passes controversial counter-terrorism law. `http://www.reuters.com/article/us-china-security-idUSKBN0UA07220151228`. Accessed: 2015-12-28.

[2] Lavabit email service abruptly shut down citing government interference. `http://www.theguardian.com/technology/2013/aug/08/lavabit-email-shut-down-edward-snowden`. Accessed: 2015-12-22.

[3] Bernstein, D. J., Lange, T., and Schwabe, P. The security impact of a new cryptographic library. Proceedings of LatinCrypt (2012).

[4] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and Thayer, R. OpenPGP Message Format. RFC 4880 (Proposed Standard), Nov. 2007. Updated by RFC 5581.

[5] Levison, L., and Watt, S. Def con 22 - dark mail. `https://www.youtube.com/watch?v=TWzvXaxR6us`. Accessed: 2015-12-22.

[6] Postel, J. Simple Mail Transfer Protocol. RFC 821 (INTERNET STANDARD), Aug. 1982. Obsoleted by RFC 2821.

[7] Ramsdell, B. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification. RFC 3851 (Proposed Standard), July 2004. Obsoleted by RFC 5751.