Email

Michalis Polychronakis

Stony Brook University
**Email Overview**

**MUA:** Mail User Agent  
Thunderbird, Apple Mail, …

**MSA:** Mail Submission Agent  
SMTP (port 587)  
Often same as initial MTA

**MTA:** Mail Transfer Agent  
SMTP (port 25)

**MDA:** Mail Delivery Agent  
IMAP (port 143), POP3 (port 110), local, …

Typical flow:  
MUA → MSA → MTA → … → MTA → MDA → MUA

©2016 OnMyWork Inc (www.OnMyWork.com) with many thanks to the GNOME project (www.gnome.org) for the images
SMTP Transport Example

S: 220 smtp.example.com ESMTP Postfix
C: HELO relay.example.org
S: 250 Hello relay.example.org, I am glad to meet you
C: MAIL FROM:<bob@example.org>
S: 250 Ok
C: RCPT TO:<alice@example.com>
S: 250 Ok
C: RCPT TO:<theboss@example.com>
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: From: "Bob Example" <bob@example.org>
C: To: "Alice Example" <alice@example.com>
C: Cc: theboss@example.com
C: Date: Tue, 15 January 2008 16:02:43 -0500
C: Subject: Test message
C:
C: Hello Alice.
C: This is a test message with 5 header fields and 4 lines in the message body.
C: Your friend,
C: Bob
C: .
S: 250 Ok: queued as 12345
C: QUIT
S: 221 Bye
Email/Messaging Security and Privacy Goals

Protect message content

Fight spam

Fight phishing

(future lecture: social engineering)

Verify communicating parties’ identities

Hide communication patterns

(future lecture: anonymity)
Who can read my email?

Adversaries with local or remote access to my devices

Intruders, spouse, system administrator, …

Malware, stolen credentials, physical access, …
Who can read my email?

Adversaries with local or remote access to MTAs and other intermediary servers

System administrators, other insiders, intruders, LEAs, …
Who can read my email?

*Adversaries with access to any intermediate network*

System administrators, other insiders, intruders, LEAs, …

Passive eavesdropping, MitM, DNS poisoning, …
Confidentiality Threats Recap:

Stored messages

- Compromised system (either local user machine or remote email server)
  Malware, intruder, insider, stolen/lost device, …

- Compromised authentication
  Password theft, phone unlock, …

Messages in transit

- Eavesdropping and interception

Displayed messages

- Screendump, reflections, shoulder surfing, …
Securing Email Transit

These days encryption is mandatory for client-to-server email transmission and retrieval

MUA ➔ MSA: STARTTLS (port 587/25), SMTPS (port 465)

MDA ➔ MUA: POP3S (port 995), IMAPS (port 993)

```
mikepo@capcom:~> nc smtp.gmail.com 25
220 mx.google.com ESMTP i185sm2356739qhc.49 - gsmtp
HELO foo.example.com
250 mx.google.com at your service
MAIL FROM:<mikepo@example.com>
530 5.7.0 Must issue a STARTTLS command first.
```

MTA ➔ MTA relaying: *a different story...*
**STARTTLS: Opportunistic Encryption**

Legacy MTAs may not support TLS
   - Fail-open design is necessary

MTAs do their best to deliver messages
   - A recipient MTA may present a self-signed cert (common in antispam/AV systems)
   - There is no PKI for email...

MitM is trivially easy
   - STARTTLS command is sent over a plaintext channel (!)
   - Analogous to SSL stripping, but in this case the client has no indication that downgrade has happened
   - Just assumes that the receiving MTA does not support TLS

Message interception is still possible
   - Better than nothing: bulk passive eavesdropping not possible
I want to STARTTLS

mikepo@capcom:~> nc aspmx.l.google.com 25
220 mx.google.com ESMTP h126si17458667qhh.29 - gsmtp
EHLO foo.example.com
250-mx.google.com at your service, [128.59.23.41]
250-SIZE 157286400
250-8BITMIME
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-CHUNKING
250 SMPTUTF8
STARTTLS
220 2.0.0 Ready to start TLS
<TLS Handshake>
I want to STARTTLS

mikepo@capcom:~> nc aspmx.l.google.com 25
220 mx.google.com ESMTP h126si17458667qhh.29 - gsmtp
EHLO foo.example.com
250-mx.google.com at your service, [128.59.23.41]
250-SIZE 157286400
250-8BITMIME
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-CHUNKING
250 SMTPUTF8
STARTTLS
220 2.0.0 Ready to start TLS
<TLS Handshake>

Can be stripped off by a MitM attacker
~60% of all messages sent via encrypted connection

Only ~30% pass strict validation (mostly due to self-signed certs)
Facebook STARTTLS Study: August 2014

~95% of outgoing messages encrypted with PFS and strict certificate validation

Mostly due to changes by big recipient networks (Microsoft, Yahoo)

How much email was encrypted in transit?

Generally speaking, use of encryption in transit increases over time, as more providers enable and maintain their support. Factors such as varying volumes of email may explain other fluctuations.

Outbound

84%
Messages from Gmail to other providers.

Inbound

73%
Messages from other providers to Gmail.
A tiny GUI change prompted many networks to deploy STARTTLS
Inbound email encryption: 99%

Start: 12/31/2013
End: 4/15/2024

Mar 3, 2024
Inbound emails: 99%
Defending against MitM

STARTTLS stripping is not the only way to intercept email

DNS MX record poisoning: spoofed MX response

  Compromised name server, MotS DNS poisoning, …
  Messages are diverted through the attacker’s mail server

DANE (DNS-based Authentication of Named Entities)

  Allow X.509 certs to be bound to DNS names through DNSSEC
  Provides a way to authenticate TLS clients/servers without a CA
  Enables downgrade-resistant TLS: advertise support for secure SMTP via a TLSA record

MTA-STS (MTA Strict Transport Security – RFC 8461)

  Allows recipient domains to tell senders whether they support TLS, how MTAs should validate certificates, and what to do if TLS negotiation fails
  Client-side policy cache provides TOFU-like protection (similar to HSTS for HTTPS)
Gmail making email more secure with MTA-STS standard
April 10, 2019

Posted by Nicolas Lidzborski, Senior Staff Software Engineer, Google Cloud and Nicolas Kardas, Senior Product Manager, Google Cloud

We’re excited to announce that Gmail will become the first major email provider to follow the new SMTP MTA Strict Transport Security (MTA-STS) RFC 8461 and SMTP TLS Reporting RFC 8460 internet standards. Those new email security standards are the result of three years of collaboration within IETF, with contributions from Google and other large email providers.

SMTP alone is vulnerable to man-in-the-middle attacks

Like all mail providers, Gmail uses Simple Mail Transfer Protocol (SMTP) to send and receive mail messages. SMTP alone only provides best-effort security with opportunistic encryption, and many SMTP servers do not prevent certain types of malicious attacks intercepting email traffic in transit.
I don’t like SPAM!
Spam Sources

Commercial entities

  Legitimate or “gray” businesses, advertisers, …

Spammers’ own hosts or open relays $\rightarrow$ easily blocked

Botnets

  Abuse of ISPs and webmail providers
  Abuse of legitimate user email accounts
  Address harvesting from users’ address books

Beyond email

  *Fraudulent messages*: Facebook, Twitter, Yelp, Amazon, online comments, forum messages, Apple/Google Store, …
  *Fraudulent activities*: likes, retweets, clicks, app store rankings, fake reviews, …
Email Spam Lifecycle

Gathering addresses
  Valid, actively used addresses are precious
  Stolen address books, web crawling, black market, …

Message content
  Advertising, 419 scams, fraud, phishing, malware, …
  Anti-spam filter evasion: content obfuscation

Spam email delivery
  Valid accounts: newly created (sweatshops), hijacked ones, …
  Fake social media accounts “primed” over time
  Open relays/proxies (not common anymore)
  Malware: most spam comes from infected machines/botnets
Email Address Protection

Keep it safe from automated address harvesting crawlers

Munging: username [at] example.com

Image instead of text

CAPTCHAs

... 

Limited effectiveness

Leaks, breaches, vendors, ...
Fighting Spam

Content-based filtering
- False positives vs. false negatives
- Local vs. cloud-based

Block lists
- IPs/domains of known spammers, open relays, zombie machines, hosts that shouldn’t be sending emails (e.g., ISP DHCP pools), …

Honeypots
- Relays, proxies, spamtraps (fake email addresses)

Outbound filtering (block port 25)
- SMTP authentication is now mandatory by most ISPs

Email authentication
Content-based Filtering

Machine learning

- Training with labeled “spam” and “ham” messages
- Feedback from user activities (e.g., “not spam” button)

Rule-based systems

- Signatures, regular expressions, patterns, …
- Certain keywords, phrases, unusual text, …
- Example: SpamAssassin

Spam authors try to evade filters

- V1agra, Via'gra, Vi@gra, vi*gra, Viagra
- Intentional spelling mistakes, symbols, weird punctuation, …
- Continuous arms race
- Example: attackers started using images, defenders started using OCR
False positives are a challenging problem
Personal example: Google’s own message classified as spam by Gmail

Important update on Chrome Supervised Users

Google Chrome <noreply-googlechrome@google.com>
to me →

Why is this message in Spam? It's similar to messages that were detected by our spam filters. Learn more

Images are not displayed. Display images below

Important update on Chrome Supervised Users

Hi Michelle,

We’re writing to you because you created a Chrome Supervised User in the past. Since we launched Chrome Supervised Users in beta preview over four years ago, Chrome and the way we use computing devices have evolved significantly. We've learned a lot in these four years, and heard feedback about how we can improve the experience for you and your children. Based on this feedback, we are working on a new set of Chrome OS supervision features specifically for the needs of families to launch later this year.

https://twitter.com/polychronakis/status/950873453648850944
DNSBL Filtering

DNS Block List

DB queried by mail servers to check the reputation of the origin of incoming email
IP addresses, domain names, and other information compiled as a DNS zone

DNS-based

Easy to query
Light on bandwidth/resources

© Spamhaus - https://www.spamhaus.org/whitepapers/dnsbl_function/
False positives, IP addresses change owners, …
SPF: Origin Authentication

SMTP allows anyone to send an email with an arbitrary “From” address

“Envelop” sender: domain included in HELO and MAIL FROM commands

Sender Policy Framework (SPF)

DNS TXT record pointing to the hosts that are allowed to send email from the domain

Receiving SMTP servers compare the IP address attempting to send an email with the allowed (by SPF) addresses of the domain provided in the SMTP envelope

Helps block spam at it source: cannot send spoofed emails from non-authorized IPs

mikepo@styx:~> dig google.com TXT
;; ANSWER SECTION:
google.com.          3599    IN     TXT   "v=spf1 include:_spf.google.com ~all"
DKIM: Email Validation

DomainKeys Identified Mail (DKIM): digitally sign some email headers and message body

Allows the recipient to verify that
  The message is sent from the domain it claims to be sent from
  The message has not been tampered with

Domain’s public key is stored in a DNS TXT record

X-Google-DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed; d=1e100.net; s=20161025;
h=x-gm-message-state:mime-version:from:date:message-id:subject:to;
bh=0BSnrwLTQ7KblIwINxoPN40a/K5PZCIV8atL6a1Dvg=;
b=Nch9yEorgibAjkh90ukDL6SU0FYn70qP6AMSFFpLO+w3iroMoVdKIjKk8Cv6Gc1TW ...

mikepo@styx:~> dig 20161025._domainkey.1e100.net TXT
;; ANSWER SECTION:
20161025._domainkey.1e100.net. 21599 IN TXT "k=rsa;
p=MIIBIjANBgkqhkiG9w0BAQEFAAOCQA8AMIIBCgKCAQEAnv6+Txyz+S Ec7mT719QQt0j6g2MjpErYUGVrRGGc7f5rmE...
**SPF + DKIM = DMARC**

*Domain-based Message Authentication, Reporting, and Conformance*

- Standardizes how email receivers perform email authentication using SPF and DKIM
- Tells receivers what to do if neither of those authentication methods passes
  - (possible actions: mark as junk, or reject the message)

*DMARC policies are published as DNS TXT records*

```bash
mikepo@styx:~/> dig _dmarc.google.com TXT
;; ANSWER SECTION:
_dmarc.google.com. 299 IN TXT "v=DMARC1; p=reject;
rua=mailto:mailauth-reports@oogle.com"
```
DMARC Email Authentication Process

http://dmarc.org/overview/
Recap: SPF, DKIM, DMARC

**SPF** validates MAIL FROM vs. its source server (“envelope” information)
Prevents spammers from sending email on behalf of a domain from other IP addresses

**DKIM** cryptographically signs a message’s headers and body
Ensures a message from a specific domain was indeed authorized by the owner of that domain
Ensures the message content is authentic and has not been altered in transit

**DMARC** specifies how emails that fail SPF+DKIM should be treated
Do nothing (just log), quarantine (place into spam/junk folder), reject

*Not effective against spammers who*
Use their own domains
Use legitimate email services (e.g., Gmail)
Are legitimate users of (or have access to) the same domain as the victim

Good for allowlisting/verifying email from trusted sources (.gov, banks, …)
End-to-End Email Encryption

Two major standards: PGP and S/MIME (similar, but incompatible)
   Both rely on public key cryptography
   Both support signing and/or encryption
   Main difference: how certificates are signed

Typical workflow
   Encrypt message with a random symmetric key
   Encrypt symmetric key with the public key(s) of recipient(s)
   Digitally sign a hash of the message

Metadata still in the clear (!)
   Email headers, appended “Received:” records, subject line
Pretty Good Privacy

De facto standard for encrypted email

PGP (Phil Zimmermann) ➔ OpenPGP (RFC 4880)
  Gnu Privacy Guard (GPG): GPL implementation

Authentication
  Senders attach their digital signature to the message
  Receivers verify the signature using public-key cryptography

Confidentiality
  Symmetric key encryption
  Random session key generated for each message
  Session key is encrypted with recipient’s public key

Both are typically used on the same message
PGP Encryption

http://www.slideshare.net/rvenkatesh25/network-security-primer
PGP Signed Message Example

From: alice@wonderland.com
Date: Mon, 16 Nov 1998 19:03:30 -0600
Subject: Message signed with PGP
MIME-Version: 1.0
Content-Type: text/plain; charset=US-ASCII
Content-Transfer-Encoding: 7bit
Content-Description: "cc:Mail Note Part"

-----BEGIN PGP SIGNED MESSAGE-----

Bob,

This is a message signed with PGP, so you can see how much overhead PGP signatures introduce. Compare this with a similar message signed with S/MIME.

Alice

-----BEGIN PGP SIGNATURE-----
Version: PGP for Personal Privacy 5.0
Charset: noconv

iQCVAWUBM+oTwFcsAarXHFeRAQEesJgp/X3noGN57U/6XYygOFjSY5lTpvAduPZ8M
aIFalUKCNuLGLGxntsbwRlDWLtCeWG3k+7zXDFx4YxzUcofG3n0QaT1k8b3nxADL0
O/E1vC/k8zJ6aGaPLB7rTlizamG0t5n6/08rPwwVkrB803tmT8UNMAUCgoM02d6HX
rKvnc2aBPFI=
=mUaH
-----END PGP SIGNATURE-----

http://www.slideshare.net/rvenkatesh25/network-security-primer
PGP Additional Features

Compression

Sign ➔ Compress ➔ Encrypt
Compression after encryption is pointless (no redundancy)
Signature does not depend on the compression algorithm

Email Compatibility

Ciphertext contains arbitrary 8-bit octects
Some email systems may interpret some of them as control commands
Solution: base64 encoding (33% space overhead)

Segmentation

Transparent message segmentation and reassembly for very large messages
Segments mailed separately
Encrypted Email: Two Main Challenges

Public key authenticity
Assurance that a public key is correct and belongs to the person or entity claimed

   Ensure it has not been tampered with or replaced by an attacker

Public key discovery
How can we find the public key of a person/entity?

   Especially the very first time we need to contact them
PGP: Web of Trust

Entirely decentralized authentication

No need to buy certs from CAs: users create their own certificates

Users validate other users’ certificates, forming a “web of trust”

No trusted authorities: trust is established through friends (yay! key signing parties!)

Main problems

Privacy issues: social graph metadata

Bootstrapping: new users are not readily trusted by others

When opinions vary, “stronger set” wins: impersonation through collusion/compromised keys

Scalability: WoT for the whole world?
Hey, I just got home from the party with the IRC folks?

Yeah.

How was it?

Got too drunk. I screwed up, bad.

What happened?

There was a girl. No idea who she was. Don't even know her name. I was too drunk to care.

And what, you slept with her?

No. I signed her public key.

Shit, man.
Finding Public Keys

Public PGP key servers

- pgp.mit.edu
- keyserver.pgp.com

Cache certificates from received emails

Integration with user management systems (LDAP)

Ad-hoc approaches

- List public key on home page
- Print on business card
- Exchange through another medium on a case-by-case basis

Association with social profiles/identities

- keybase.io
MIT PGP Public Key Server

Help: Extracting keys / Submitting keys / Email interface / About this server / FAQ
Related Info: Information about PGP /

Extract a key

Search String: [ ] Do the search!

Index:  
- Verbose Index:

- Show PGP fingerprints for keys
- Only return exact matches

Submit a key

Enter ASCII-armored PGP key here:

[ ]
keybase.io/mikepo

Michalis Polychronakis

keybase.io/mikepo

8EBD 8F30 8899 8AFF

polychronakis • tweet
polychronakis • gist

mikepo has an invitation available
If you know mikepo, you can ask them for an invitation to Keybase.

Encrypt
Verify

mikepo from the command line

# first
keybase join  # if you're new, or
keybase login  # if you're not.

# then
keybase push  # if you already have a public key, or
keybase gen  # if this is all new to you

Tracking (6)
mikepo
hargikas
mstamat
gianluca_string

Trackers (6)
hargikas
kontaxis
mstamat
Keybase.io

In essence, a directory associating public keys with names

Identity established through *public signatures*

**Identity proofs:** “I am Joe on Keybase and MrJoe on Twitter”

**Follower statements:** “I am Joe on Keybase and I just looked at Chris's identity”

**Key ownership:** “I am Joe on Keybase and here's my public key”

**Revocations:** “I take back what I said earlier”

Keybase identity = sum of public identities

Twitter, Facebook, Github, Reddit, domain ownership, …

An attacker has to compromise all connected identities

The more connected identities, the harder to impersonate a user
Biggest Issue: Usability

Non-trivial setup

  PGP: users are responsible for everything

Key management

Key revocation

Public key fingerprints

Poor mail client integration

  Can lead to catastrophic failures: e.g., Enigmail+Thunderbird silent encryption failure

(Let alone key discovery and trustworthiness issues)

https://xkcd.com/1181/
Enigmail 1.7 is completely broken for my purposes.

Steps to reproduce the problem:

1) Write an email in TB.

2) Ensure “Force encryption” in Enigmail.

3) Ensure “Force signing” in Enigmail.

4) Recheck encryption and signing settings… OK.

5) Send the email.

6) Look at the received email. OOPS. It is NOT signed and NOT encrypted.

Sorry to say this so directly, but an encryption system, which CONFIRMS to the user in its graphical user interface on two different places that it will encrypt AND THEN SENDS THE EMAIL WITHOUT ANY ENCRYPTION IN PLAIN TEXT ... is just the BIGGEST IMAGINABLE CATASTROPHE.

Sorry for my profane language but there is simply no excuse for such
I've been posting my public key for 15 years now, but no one has ever asked me for it or used it for anything as far as I can tell.

Maybe I should try posting my private key instead.
Swedish media org @Aftonbladet publishes its GPG private key for a second time (first time was in 2012):

Anders Nilsson @nilssonanders
Sweden's biggest newspaper #Aftonbladet includes their private key in guide to PGP mail them (via @_zulin ) bit.ly/1FfHAOI
PSIRT PGP Key (0x33E9E596)

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: Mailvelope v1.8.0
Comment: https://www.mailvelope.com

xsFNBFm/2KMBEADbwToJMN3BCVE10oC22HgVEqNEpXzud2dfgKuy0M4tx2LDe7Gkpo6Aoww4y8bakLiIdpw5B0J/AR1VtjTIDEm50F9MRZICV0UKy5aqVa9bfAicY7nezkIJUmyLcIVMC60pqSHzoOEwy2PjzxcI4vDHSmcgf5XRe+uYld3LTVI+A/5jv326LB16bcCNts/tohW2T0LznPcCtidH84Z4tPcy3p358dZ/cD4+iM41iKy21kq/EzVNCv8aS3RUXy27SLN1VHeYml6UNKCOeHCn2s2yJxMiBCniozBZKuZw6s1g97nnq663d9mf7V30PS3zaAh30Hvmzg3BNoqzy2dAEU/JDUBhIao+x9VF9Z3POoC8yst0RgyUm/2t3TBTaH+DnfsUBoqSU2T0n8x2R1Fxw2YZNCtuKu5J0vPqtRBf13DsyTD7LDps62nqhpVb34epwukq1K0TRU9mB4E0q+cNFR32pNiAK+jHOb/3/UJwCJyPvju2/3Gwqddq8H+QqVcNm8vIGNQZQW30WqNH/UFoh3RPJ+WqNdqB48NMQBq8I4aA4u8MqoObd/zrtVXkAyhbbzI00925nJfPyuxmhIcOtkKen18dzeFBaB81jRyuIMCNjQ0GqJ+Q8TJyEesNdK/qBDH51hKRCSzKHDL+Raz3/z+1/FIFwARAQBArzRBZG9iZSBQU015VCA89cNpcnRRAYWRvYmUyU297PsLBewQAQaGhLaUWcb/TrvUJXebEzbAYLCqgHzAiJEBIAd8Kvh3YWBBUIA0DgPgIBAhkBAhDAh4BADk2A/6+fPFP2q4VMLiPZ8ZPoqPR/1X1Z7R1yBwFyW0W0XiL2EeD507HQT6NNMOW15JrwsFvQOWTa1u96SHRoU1KGTSSR6oQ5nHnX4DrCBHSKm0MS68pUz88wYON4Upe9V9PuuueU040s7VHnE5gBQpQurtv80wQc4uUtwmFnjN7N40H0++2910JP66B6+kMuQyG4swmxzHvlljgGMHsc0s/Bu13W+n5txLM7N5j5jtCjNXRtGmstdmRFbOWWosE2iwN0UKLCJuYHap36H9x8R13wpr2e0QgqKIqRq73D4UAcqsp80

-----END PGP PUBLIC KEY BLOCK-----
**S/MIME**

Based on standard X.509 certificates

- Analogous operation to TLS: trusted CA sign certificates
- Traditional PKI

Uses MIME to include cryptographic information in the message

- Multipurpose Internet Mail Extensions: extends the format of email messages to support binary attachments, and text in non-ASCII character sets

Works well within corporations

- Certificate distribution through the existing Active Directory infrastructure

Built-in support in most modern email clients

- Seamless interoperability between them
S/MIME Signed Message Example

From: alice@wonderland.com
Date: Mon, 16 Nov 1998 19:03:08 -0600
Subject: Message signed with S/MIME
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="simple boundary"

--simple boundary
Content-Type: text/plain; charset=US-ASCII
Content-Transfer-Encoding: 7bit
Content-Description: "cc:Mail Note Part"

Bob,

This is a message signed with S/MIME, so you can see how much overhead S/MIME signatures introduce. Compare this with a similar message signed with PGP.

Alice

--simple boundary
Content-Type: application/octet-stream; name="smime.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7s"

MITQwYJKoZIhvcNAQcCoITQNDiCedECAAQExCzAJBgUrDgMCGgUAMAAsGCSqGSIb3DQEHAaCCDww
ggGMIIJLADA4gECAhBBQRx9a+Dx0FHXqQVHqFMA0GCSqGSIb3DQEDEBBIAGxETAPBgNVBAcT
CEIudGYvYbVKpVMRCwFDQVQKEwWZXMjU2LnhwSW5jLjE0MDI1M1UEtXMrVnYmVpX24ZqGQ2xh
3MgHQBQSAtle1luG12aWRIYwgwU3Vic2NyaWJlcm1AcFw05NgwMjcwMDMwMTBAMDBaFW05ODA2Mjcy
Mu5NT1aMi1BfzERMA8GAAWhMSW50ZXJzU2l0cm1TaWduLCBdUmNsMTQw
MgYDVQQLEy1wZXMjU2LnhwSW5kaXZpZmluZyBhMC8wCAYDVQQGEwJNMCowMDk0MDEyMDk0NjIwNTYwNzEz
MIIBTzIaMjEgQXV0dFQgQ8BzASQwMB0GCSqGSIb3DQEJARYXc= 63

http://www.slideshare.net/rvenkatesh25/network-security-primer
End-to-End vs. Cloud-to-Cloud

IMAP: one of the oldest “cloud” services!

- Keep messages on the server
- Conveniently access them from multiple devices (no file synchronization needed)

Useful modern cloud-based email features

- Powerful and rapid search, collaborative SPAM filtering, …
- Need access to the plaintext (!) Gmail cannot index or filter encrypted messages

Tradeoff: privacy vs. convenience

- Active research on searchable encryption
Encrypted Webmail?

Several recent efforts have focused on transparently combining the convenience of webmail with PGP encryption

- Is this really possible in a secure way?

JavaScript crypto is not a good idea

- Secure JS code delivery?
- Secure key storage?
- Secure runtime (it’s a web browser!)?

Google end-to-end: implement cryptographic functionality as part of a browser extension

- More control, but still not trivial
- After initial excitement, it seems the effort has been abandoned
My Fellow Users,

I have been forced to make a difficult decision to become complicit in crimes against the American people or walk away from nearly ten years of hard work by shutting down Lavabit. After significant soul searching, I have decided to suspend operations. I wish that I could legally share with you the events that led to my decision. I cannot. I feel you deserve to know what’s going on—the first amendment is supposed to guarantee me the freedom to speak out in situations like this. Unfortunately, Congress has passed laws that say otherwise. As things currently stand, I cannot share my experiences over the last six weeks, even though I have twice made the appropriate requests.

What’s going to happen now? We’ve already started preparing the paperwork needed to continue to fight for the Constitution in the Fourth Circuit Court of Appeals. A favorable decision would allow me to resurrect Lavabit as an American company.

This experience has taught me one very important lesson: without congressional action or a strong judicial precedent, I would _strongly_ recommend against anyone trusting their private data to a company with physical ties to the United States.

Sincerely,
Ladar Levison
Owner and Operator, Lavabit LLC

Defending the constitution is expensive! Help us by donating to the Lavabit Legal Defense Fund here.
Lavabit: “so secure that even our administrators can’t read your e-mail”

But they could, if they wanted to…

“Basically we generate public and private keys for the user and then encrypt the private key using a derivative of the plain text password. We then encrypt user messages using their public key before writing them to disk.”

“Because we need the plain text password to decrypt a user’s private key, we don’t support secure password authentication. We decided to support SSL instead (which encrypts everything; not just the password).”
Maybe rethink email altogether?

Secure messaging apps offer many benefits

- True end-to-end encryption: the provider cannot read message content
- User-friendly verification of contacts’ identities
- Forward secrecy: past communications remain secure even if private keys are stolen
- *No spam!* Only approved contacts can send messages

Best option: **Signal**

- Double Ratchet Algorithm (precursor: [OTR protocol](#))
- **Privacy-preserving contact discovery**

OK alternatives (closed-source): WhatsApp (uses Signal protocol), iMessage

Metadata is still there!

- Signal is actively trying to minimize it
Grand jury subpoena for Signal user data (2016)

Dear Sir/Madam:

You have been served with a subpoena issued in connection with a criminal investigation being conducted in this District. That subpoena directs you to produce certain records on 7/14/2016 before the grand jury in Alexandria, Virginia.

<table>
<thead>
<tr>
<th>Account</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Last connection date: 1454198400000 Unix millis</td>
</tr>
<tr>
<td></td>
<td>Account created: 145347522063 Unix millis</td>
</tr>
</tbody>
</table>

https://signal.org/bigbrother/eastern-virginia-grand-jury/
Building end-to-end security for Messenger

By Jon Millican, Reed Riley

- We are beginning to upgrade people's personal conversations on Messenger to use end-to-end encryption (E2EE) by default.
- Meta is publishing two technical white papers on end-to-end encryption:
  - Our Messenger end-to-end encryption whitepaper describes the core cryptographic protocol for transmitting messages between clients.
  - The Labyrinth encrypted storage protocol whitepaper explains our protocol for end-to-end encrypting stored messaging history between devices on a user's account.

Today, we're announcing that we've begun to upgrade people's personal conversations on Messenger to use E2EE by default. Our aim is to ensure that everyone's personal messages on Messenger can only be accessed by the sender and the intended recipients, and that everyone can be sure the messages they receive are from an authentic sender.

This is the most significant milestone yet for this project, which began in earnest after Mark Zuckerberg outlined his vision for it in 2019. Bringing E2EE to Messenger has been a complex process, with every feature and product goal revealing further challenges that must be overcome to ensure the privacy and security of our users.