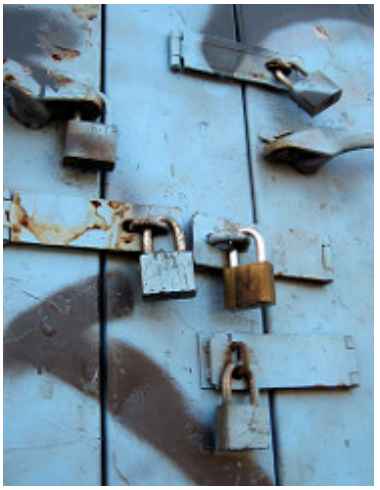
Matthew Green in messaging, privacy August 13, 2014July 29, 2016 2,592 Words

What's the matter with PGP?



Last Thursday, Yahoo announced their plans to support end-to-end encryption using a fork of Google's end-to-end email extension (https://code.google.com/p/end-to-end/). This is a Big Deal. With providers like Google and Yahoo onboard, email encryption is bound to get a big kick in the ass. This is something email badly needs (http://www.newyorker.com/tech/elements/the-daunting-challenge-of-secure-e-mail).

So great work by Google and Yahoo! Which is why following complaint is going to seem awfully ungrateful. I realize this and I couldn't feel worse about it.

As transparent and user-friendly as the new email extensions are, they're fundamentally just re-implementations of OpenPGP (http://en.wikipedia.org/wiki/Pretty_Good_Privacy) — and non-legacy-compatible ones, too. The problem with this is that, for all the good PGP has done in the past, it's a model of email encryption that's fundamentally broken.

It's time for PGP to die.

In the remainder of this post I'm going to explain why this is so, what it means for the future of email encryption, and some of the things we should do about it. Nothing I'm going to say here will surprise anyone who's familiar with the technology — in fact, this will barely be a technical post. That's because, fundamentally, most of the problems with email encryption aren't hyper-technical problems. They're still baked into the cake.

Background: PGP

Back in the late 1980s a few visionaries realized that this new 'e-mail' thing was awfully convenient and would likely be the future — but that Internet mail protocols made virtually no effort to protect the content of transmitted messages. In those days (and still in these days

(https://www.eff.org/deeplinks/2014/06/new-gmail-data-shows-rise-backbone-email-encryption)) email transited the Internet in cleartext, often coming to rest in poorly-secured mailspools.

This inspired folks like Phil Zimmermann to create tools to deal with the problem. Zimmermann's PGP was a revolution. It gave users access to efficient public-key cryptography (http://en.wikipedia.org/wiki/Public-key_cryptography) and fast symmetric ciphers in package you could install on a standard PC. Even better, PGP was compatible with legacy email systems: it would convert your ciphertext into a convenient ASCII armored (http://en.wikipedia.org/wiki/Binary-to-text_encoding) format that could be easily pasted into the sophisticated email clients of the day — things like "mail", "pine" or "the Compuserve e-mail client".
It's hard to explain what a big deal PGP was. Sure, it sucked badly to use. But in those days, *everything* sucked badly to use. Possession of a PGP key was a badge of technical merit. Folks held key signing parties (http://en.wikipedia.org/wiki/Key_signing_party). If you were a geek and wanted to discreetly share this fact with other geeks, there was no better time to be alive.

We've come a long way since the 1990s, but PGP mostly hasn't. While the protocol has evolved technically — IDEA (http://en.wikipedia.org/wiki/International_Data_Encryption_Algorithm) replaced BassOMatic (http://en.wikipedia.org/wiki/BassOmatic), and was in turn replaced by better ciphers — the fundamental concepts of PGP remain depressingly similar to what Zimmermann offered us in 1991. This has become a problem, and sadly one that's difficult to change.
Let's get specific.

PGP keys suck

Before we can communicate via PGP, we first need to exchange keys. PGP makes this downright unpleasant. In some cases, dangerously so.
Part of the problem lies in the nature of PGP public keys themselves. For historical reasons they tend to be large and contain lots of extraneous information, which it difficult to print them a business card or manually compare. You can write this off to a quirk of older technology, but even modern elliptic curve implementations (http://tools.ietf.org/html/rfc6637) still produce surprisingly large keys.

(https://matthewdgreen.files.wordpress.com/2014/08/ba0d3-keys.png)

Three public keys offering roughly the same security level. From top-left: (1) Base58-encoded Curve25519 public key used in miniLock (https://minilock.io/). (2) OpenPGP 256-bit elliptic curve public key format. (3a) GnuPG 3,072 bit RSA key and (3b) key fingerprint.

Since PGP keys aren't designed for humans, you need to move them electronically. But of course humans still need to verify the *authenticity* of received keys, as accepting an attacker-provided public key can be catastrophic (http://en.wikipedia.org/wiki/Man-in-the-middle_attack).

PGP addresses this with a hodgepodge of key servers (http://en.wikipedia.org/wiki/Key_server_(cryptographic)) and public key fingerprints (http://en.wikipedia.org/wiki/Public_key_fingerprint). These components respectively provide (untrustworthy) data transfer and a short token that human beings can manually verify. While in theory this is sound, in practice it adds complexity, which is always the enemy of security.

Now you may think this is purely academic. It's not. It can bite you in the ass.

Imagine, for example, you're a source looking to send secure email to a reporter at the Washington Post. This reporter publishes his fingerprint via Twitter (https://twitter.com/bartongellman/status/412298520935153664), which means most obvious (and recommended (https://twitter.com/ioerror/status/496369562276069378)) approach is to ask your PGP client to retrieve the key by fingerprint from a PGP key server. On the GnuPG command line can be done as follows:

```
Computer-3:Desktop mgreen$ gpg --recv-key "1392043726605D0ED556C8A405F1287DC38C85C0"
gpg: requesting key C38C85C0 from hkp server keys.gnupg.net
gpg: key C38C85C0: public key "Barton Gellman <otr@riseup.net>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

(https://matthewdgreen.files.wordpress.com/2014/08/58cb1-recvkeysworks.png)

Now let's ignore the fact that you've just leaked your key request to an untrusted server via HTTP. At the end of this process you should have the right key with high reliability. Right?

Except maybe not: if you happen to do this with GnuPG 2.0.18 — one version off from the very latest GnuPG — the client *won't actually bother to check the fingerprint of the received key* (http://bugs.gnupg.org/gnupg/issue1579). A malicious server (or HTTP attacker) can ship you back the wrong key and you'll get no warning. This is fixed in the very latest versions of GPG but… Oy Vey.

(https://matthewdgreen.files.wordpress.com/2014/08/0f936-keyid.png)

PGP Key IDs are also pretty terrible, due to the short length and continued support for the broken V3 key format.

You can say that it's unfair to pick on all of PGP over an implementation flaw in GnuPG, but I would argue it speaks to a fundamental issue with the PGP design. PGP assumes keys are too big and complicated to be managed by mortals, but then in practice it practically begs users to handle them anyway. This means we manage them through a layer of machinery, and it happens that our machinery is far from infallible.

Which raises the question: why are we bothering with all this crap infrastructure in the first place. If we *must* exchange things via Twitter, why not simply exchange *keys?* Modern EC public keys are tiny. You could easily fit three or four of them in the space of this paragraph. If we must use an infrastructure layer, let's just use it to shunt all the key *metadata* around.

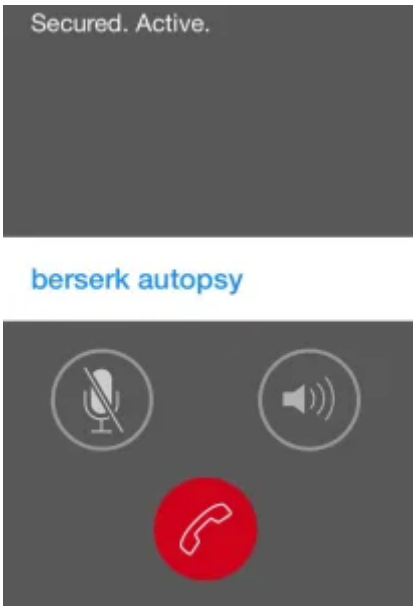
PGP key management sucks

Manual key management is a mug's game. Transparent (or at least *translucent*) key management is the hallmark of every successful (http://techcrunch.com/2014/02/27/apple-explains-exactly-how-secure-imessage-really-is/) end-to-end secure (http://www.theverge.com/2014/7/29/5945547/signal-brings-painless-encrypted-calling-whisper-systems-moxie-marlinspike) encryption system.

Now often this does involve some tradeoffs — e.g.,, the need to trust a central authority to distribute keys — but even this level of security would be lightyears better than the current situation with webmail.

If you can't trust Phil, who can you trust?

To their credit, both Google and Yahoo have the opportunity to build their own key management solutions (at least, for those who trust Google and Yahoo), and they may *still* do so in the future (https://twitter.com/alexstamos/status/497591770645942275). But today's solutions don't offer any of this, and it's not clear when they will. Key management, not pretty web interfaces, is the real weakness holding back widespread secure email.



ZRTP authentication string, as used in Signal (https://whispersystems.org/).

For the record, classic PGP does have a solution to the problem. It's called the "web of trust (http://en.wikipedia.org/wiki/Web_of_trust)", and it involves individuals signing each others' keys. I refuse to go into the problems with WoT because, frankly, life (https://lists.torproject.org/pipermail/tor-talk/2013-September/030235.html) is too short. The TL;DR is that 'trust' means different things to you than it does to me. Most OpenPGP implementations do a lousy job of presenting any of this data to their users anyway.

The lack of transparent key management in PGP isn't unfixable. For those who don't trust Google or Yahoo, there are experimental systems like Keybase.io (https://keybase.io/) that attempt to tie keys to user identities. In theory we could even exchange our offline encryption keys through voice-authenticated channels using apps like OpenWhisperSystems' Signal

(https://itunes.apple.com/us/app/signal-private-messenger/id874139669?mt=8). So far, nobody's bothered to do this — all of these modern encryption tools are islands with no connection to the mainland. Connecting them together represents one of the real challenges facing widespread encrypted communications.

No forward secrecy

Try something: go delete some mail from your Gmail account. You've hit the archive button. Presumably you've also permanently wiped your Deleted Items folder. Now make sure you wipe your browser cache and the mailbox files for any IMAP clients you might be running (e.g., on your phone). Do any of your devices use SSD drives (http://arstechnica.com/security/2011/03/ask-ars-how-can-i-safely-erase-the-data-from-my-ssd-drive/)? Probably a safe bet to securely wipe those devices entirely. And at the end of this Google may *still* have a copy which could be vulnerable to law enforcement request or civil subpoena.

(Let's not get into the NSA's collect-it-all policy for encrypted messages (http://www.theguardian.com/world/2013/jun/20/fisa-court-nsa-without-warrant). If the NSA is your adversary just forget about PGP.)

Forward secrecy (http://en.wikipedia.org/wiki/Forward_secrecy) (usually misnamed "*perfect* forward secrecy") ensures that if you can't destroy the ciphertexts, you can at least dispose of keys when you're done with them. Many online messaging systems like off-the-record (https://otr.cypherpunks.ca/) messaging use PFS by default, essentially deriving a new key with each message volley sent. Newer 'ratcheting' systems like Trevor Perrin's Axolotl (https://github.com/trevp/axolotl/wiki) (used by TextSecure (https://play.google.com/store/apps/details?id=org.thoughtcrime.securesms&hl=en)) have also begun to address the offline case.

Adding forward secrecy to asynchronous offline email is a much bigger challenge, but fundamentally it's at least *possible* to some degree. While securing the initial 'introduction' message between two participants may be challenging*, each subsequent reply can carry a new ephemeral key to be used in future communications. However this requires breaking changes to the PGP protocol and to clients — changes that aren't likely to happen in a world where webmail providers have doubled down on the PGP model.

The OpenPGP format and defaults suck

Poking through a modern OpenPGP implementation (https://www.gnupg.org/) is like visiting a museum of 1990s crypto. For legacy compatibility reasons, many clients use old ciphers like CAST5 (http://en.wikipedia.org/wiki/CAST-128) (a cipher that predates the AES competition (http://competitions.cr.yp.to/aes.html)). RSA encryption uses padding that looks disturbingly like PKCS#1v1.5 (http://tools.ietf.org/html/rfc2313) — a format that's been relentlessly exploited in the past (https://blog.cryptographyengineering.com/2012/06/bad-couple-of-years-for-cryptographic.html). Key size defaults don't reach the 128-bit security level. MACs (http://en.wikipedia.org/wiki/Message_authentication_code) are optional. Compression is often on by default. Elliptic curve crypto is (still!) barely supported (https://code.google.com/p/gnupg-ecc/).

Most of these issues are *not* exploitable unless you use PGP in a non-standard way, e.g., for instant messaging (http://box.matto.nl/gnupgjabber.html) or online applications. And some people do use PGP this way.

But even if you're just using PGP just to send one-off emails to your grandmother, these bad defaults are *pointless and unnecessary.* It's one thing to provide optional backwards compatibility for that one friend who runs PGP on his Amiga. But few of *my* contacts do — and moreover, *client versions are clearly indicated in public keys.*** Even if these archaic ciphers and formats aren't exploitable today, the current trajectory guarantees we'll still be using them a decade from now. Then all bets are off.

If Will Smith looked like this when your cryptography was current, you need better cryptography.

On the bright side, both Google and Yahoo seem to be pushing towards modern implementations that break compatibility with the old. Which raises a different question. If you're going to break compatibility with most PGP implementations, why bother with PGP at all?

Terrible mail client implementations

This is by far the worst aspect of the PGP ecosystem, and also the one I'd like to spend the least time on. In part this is because UX isn't technically PGP's problem; in part because the experience is inconsistent between implementations, and in part because it's inconsistent between *users:* one person's 'usable' is another person's technical nightmare.

But for what it's worth, many PGP-enabled mail clients make it ridiculously easy to send confidential messages with encryption turned off, to send unimportant messages with encryption turned on, to accidentally send to the wrong person's key (or the wrong subkey within a given person's key). They demand you encrypt your key with a passphrase, but routinely bug you to enter that passphrase in order to sign outgoing mail — exposing your decryption keys in memory even when you're not reading secure email.

Most of these problems stem from the fact that PGP was designed to retain compatibility with standard (non-encrypted) email. If there's one lesson from the past ten years, it's that people are comfortable moving *past* email. We now use purpose (http://techcrunch.com/2014/02/27/apple-explains-exactly-how-secure-imessage-really-is/)–built (https://www.facebook.com/mobile/messenger) messaging (http://www.whatsapp.com/) systems on a day-to-day basis. The startup cost of a secure-by-default environment is, at this point, basically an app store download.

Incidentally, the new Google/Yahoo web-based end-to-end clients dodge this problem by providing essentially no user interface at all. You enter your message into a separate box, and then plop the resulting encrypted data into the Compose box. This avoids many of the nastier interface problems, but only by making encryption non-transparent. This may change; it's too soon to know how.

So what should we be doing?

Quite a lot actually. The path to a proper encrypted email system isn't that far off. At minimum, any real solution needs:

- **A proper approach to key management.** This could be anything from centralized key management as in Apple's iMessage (http://images.apple.com/iphone/business/docs/iOS_Security_Feb14.pdf) — which would still be better than nothing — to a decentralized (but still usable) approach like the one offered by Signal

or OTR. Whatever the solution, in order to achieve mass deployment, keys need to be made much more manageable or else submerged from the user altogether.
- **Forward secrecy baked into the protocol.** This should be a pre-condition to any secure messaging system.
- **Cryptography that post-dates the Fresh Prince**. Enough said.
- **Screw backwards compatibility.** Securing both encrypted and unencrypted email is too hard. We need dedicated networks that handle this from the start.

A number of projects are already going in this direction. Aside above-mentioned projects like Axolotl and TextSecure — which pretend to be text messaging systems, but are really email in disguise — projects like Mailpile (https://www.mailpile.is/) are trying to re-architect the client interface (though they're sticking with the PGP paradigm). Projects like SMIMP (https://github.com/smimp/smimp_spec/blob/master/smimp_specification.md) are trying to attack this at the protocol level.*** At least in theory projects like DarkMail (https://www.darkmail.info/) are also trying to adapt text messaging (https://silentcircle.com/scimp-protocol) protocols to the email case, though details remain few and far between.

It also bears noting that many of the issues above could, in principle at least, be addressed within the confines of the OpenPGP format. Indeed, if you view 'PGP' to mean nothing more than the OpenPGP transport, a lot of the above seems easy to fix — with the exception of forward secrecy, which really does seem hard to add without some serious hacks. But in practice, this is rarely all that people mean when they implement 'PGP'.

Conclusion

I realize I sound a bit cranky about this stuff. But as they say: a PGP critic is just a PGP user who's actually *used* the software for a while. At this point so much potential in this area and so many opportunities to do better. It's time for us to adopt those ideas and stop looking backwards.

Notes:

* Forward security even for introduction messages can be implemented, though it either require additional offline key distribution (e.g., TextSecure's 'pre-keys' (https://whispersystems.org/blog/asynchronous-security/)) or else the use of advanced primitives (https://www.cs.umd.edu/~jkatz/papers/forward-enc-full.pdf). For the purposes of a better PGP, just handling the second message in a conversation would be sufficient.

** Most PGP keys indicate the precise version of the client that generated them (which seems like a dumb thing to do). However if you want to add metadata to your key that indicates which ciphers you prefer, you have to use an optional command.

*** Thanks to Taylor Hornby for reminding me of this.

73 thoughts on "What's the matter with PGP?"

Birgitta Jónsdóttir says:

August 13, 2014 at 6:22 pm
thank you!

Moschops says:
August 13, 2014 at 6:24 pm
Given that Google, Yahoo and chums have an essentially open-door policy for the NSA (and by association, GCHQ, the rest of five eyes, and probably anyone who asks nicely), why should we trust anything they do for us? Genuine question; why should we trust them in any way with encryption?

Matthew Green says:
August 13, 2014 at 6:32 pm
Like I said, if NSA and GCHQ are your concern you need to go above and beyond. That doesn't rule out centralized key management for *some* users, with the option to verify key fingerprints for the paranoid.

David Karapetyan says:
August 13, 2014 at 6:55 pm
You forgot protonmail.

Moschops says:
August 13, 2014 at 7:08 pm
Is it still paranoia now that we know for sure it's happening? ☺

Simon Nicolussi says:
August 13, 2014 at 7:13 pm
GnuPG (1.4.16/2.0.23 and above) no longer indicates its minor version in exported keys by default.

Kevin Riggle says:
August 13, 2014 at 7:25 pm
This comment has been removed by the author.

Kevin Riggle says:
August 13, 2014 at 7:26 pm
If I'm going to trust Google and Yahoo with my e-mail anyway, we might as well dispense with this PGP decentralized-web-of-trust nonsense and use a solution which builds on the existing web PKI model, which people who aren't cryptogeeks actually do use and derive value from. We've even got the protocol for it (S/MIME), we just need Google and Yahoo to support it in their webmail products.

Jason Brunette says:
August 13, 2014 at 7:30 pm
Is the darknet and it's markets safe ? They depend on PGP.

Anonymous says:
August 13, 2014 at 7:39 pm
The current version of the OTR protocol only supports 1024-bit DSA public keys, this is much weaker then PGP defaults.
http://cr.yp.to/talks/2010.04.16/slides.pdf

Many instant messages with OTR keep a history of all transmitted messages in clear text. This defeats the purpose of forward secrecy, because many IM user don't know/care about this. From practical point of view the forward secrecy is better in VoIP systems with ZRTP because most users don't keep records of the voice conversations.

Martok says:
August 13, 2014 at 8:16 pm
Now, as someone who is currently implementing an OpenPGP/RFC4880-conformant library for those applications where deploying GnuPG isn't an option, I might be a bit biased towards wanting it to survive 😉

However, it seems to me that pretty much everything you mention except (P)FS is just an implementation issue. If you look at the actual RFC, you'd be surprised how much their relatively sensible recommendations diverge from what implementors do. To add to that, GnuPG, probably the most widely used implementation (we use that thing to bootstrap Linux distros, for heavens sakes), actively violates some parts of the spec in a way that makes it less safe. That includes stupid cipher/hash preferences (cases of MUST be implemented, there's nothing saying they should be default), ignoring the receiving key's cipher preferences altogether under some circumstances and doesn't end with hard-failing on generating longer keys (read: keys that have a chance to survive the next decade unfactored). MACs are actually not really all that optional in 4880-compliant implementations – almost-mandatory MDCs coupled with the sign+encrypt-feature serve precisely that purpose.

A lot of the weird stuff mail clients do comes from the fact that they rely on GPG as their backend (hello, Enigmail!) with its odd method of specifying what to do. GnuPGs modes of operation often map directly to UI elements, even if they make little to no sense. On that occasion, why do you need to import a key in the first place? What stops your mail client from background-checking if a recipient has a key on the keyservers and offering to fetch&encrypt automatically?

Something that occured to me while implementing the basics: there's actually a nice way of doing OTR right in the RFC: nothing stops you from providing a throwaway reply pubkey *right with your message*. Of course, you probably want to sign that again to create a chain of keys from the original point of manual KEx to "now", but again, that's just semantics.

So the big issue I see is that GnuPG is utterly broken(*) and needs to die… and with it all those 3rd-Party-Implementations that try to be compatible to GPG instead of OpenPGP as written in the RFCs. OpenPGP in itself is a pretty nice format, and in most places one of the saner things to happen in crypto.

*) Don't even get me started on the sheer code "quality" there. Go ahead and try to trace execution for a simple message signature verification… the tucked-on nature of ECC is not even the worst part.

Anonymous says:
August 13, 2014 at 8:33 pm
What about LEAP email?
https://leap.se/en/services/email

grawity says:
August 13, 2014 at 8:37 pm
You're assuming the protocols, formats, algoritms used would be better. They're in practice worse than PGP… See for example Gutmann's "Godzilla Crypto Guide" or "Everything you Never Wanted to Know about PKI" or "X.509 Style Guide".

Anonymous says:
August 13, 2014 at 8:40 pm
This seems unlikely to succeed on a large scale simply because it makes webmail unsearchable. Unless they implement client-side indexing in Javascript, which would be… interesting.

Kevin Riggle says:
August 13, 2014 at 9:24 pm
Believe me, I know about X.509, and I have spent far too long staring at the output of `openssl x509 -text -noout`. It's still better than PGP, because the web PKI model makes sense to people who aren't cryptogeeks ("I trust Verisign, and Verisign verifies me, and Firefox trusts Verisign to verify me"), and the tools are usable by people who aren't cryptogeeks.

You never even have to understand X.509 or PKI to get a HTTPS web server up and running which provides reasonable security guarantees. Every CA will tell you exactly how to generate a reasonable cert and how to install it in your web server of choice.

ipgmail says:
August 13, 2014 at 9:33 pm
As an author of an independent implementation of RFC 4880, I agree with most of these points. It is very hard to simplify PGP to the point where it is NOT a burden to use. The primary reason is the key management – how to seamlessly and securely exchange public keys. There are technical as well as significant "social" usability issues that will probably prevent PGP from ever being a widely used technology.

Most users of PGP are (rightfully) paranoid and distrusting of a centralized key servers and will probably never trust yahoo or google or anyone else to manage their private keys. Because of that, every time someone wants to use PGP on a new computer or device, they have to "securely" move their private key file around and then trust the new system not to leak it. A trusted centralized key service would be great but getting enough people to trust it enough to make it useful on a large scale is a real challenge.

Anonymous says:
August 14, 2014 at 1:41 am
We're building an email provider that might address some of the problems list. It's a zero-knowledge encrypted email provider that handles key exchange and verification between our users automatically. We also use perfect forward secrecy and have a key exchange system in the pipeline that will grab keys from popular keybases for Hushmail users, I agree with you that the key exchange is one of the fundamental issues with making PGP mainstream – however its usage has surged over the past year and new email providers like ours have conquered the usability issue – since we launched Gmail and Yahoo! Mail have both announced tighter security measures, as have Apple with iMessage encryption. The big companies are listening, PGP isn't a lost cause, we just need to make it easy use. Lavaboom.com

Hauke Laging says:
August 14, 2014 at 2:34 am
Why should anyone care about the key size? Fingerprints have been on business cards for more than a decade. The miniLock ID is even longer (OK, that is going to change in the future). "miniLock will refuse weak passphrases completely" – users will love that.

Fingerprint comparison is not limited to being done manually (and if that is a problem why isn't it for miniLock?). That could easily be done with QR codes if the demand was there.

Furthermore it is quite strange to compare the size of a key to that of a certificate. As if the information in a certificate wasn't of any use and could just be thrown away…

"At the end of this process you should have the right key with high reliability. Right?"

Yes, right. The "end of the process" is not importing the key but making it valid (if the WoT doesn't do that) by verifying and signing. Fingerprints are to be compared not to be entered. I have never read the recommendation to *search* for a certificate by its fingerprint.

"PGP assumes keys are too big and complicated to be managed by mortals, but then in practice it practically begs users to handle them anyway."

That is a really strange statement.

"If we must exchange things via Twitter"

Do we have to? Is that the crypto future, relying on *Twitter*? Seriously?

Forward secrecy is not an important feature for email. People want to read their data not throw it away. It can even be quite disturbing with instant messaging. If you want forward secrecy for certain communication then use another tool. OpenPGP's lack of FS is less annoying than the other tools' inability to disable it. I don't remember anyone asking "Will there be FS in future OpenPGP versions?".

"these bad defaults" – AES256, AES192, AES, CAST5, 3DES, IDEA; SHA256, SHA1, SHA384, SHA512, SHA224. That (GnuPG 2.0.22) is bad defaults?

"Terrible mail client implementations"

That is an opinion. A fact is that even non-IT people can easily learn that. There will never be security without learning anyway.

What is "the wrong subkey within a given person's key"? Which mail client offers the selection of subkeys?

Who cares about a cached passphrase? Is that supposed to be a real protection? That would require the passphrase to resist at least a poor man's brute force attack. Who uses such a passphrase on an everyday system? Caching can be disabled. And who would use different passphrases for signing and decryption, seriously? This IS purely academic.

"Key size defaults don't reach the 128-bit security level."

Why should they? The systems these keys are used on have some "64-bit security level". Would the NSA try to brute force 100-bit keys? No. Will 112-bit keys be brute force attacked in less than 20 years? Improbable. Are those systems suitable for handling data that must be confidential over 20 years? Once again: A purely academic security gain. You like miniLock – which usually offers 111 bit entropy.

"If there's one lesson from the past ten years, it's that people are comfortable moving past email."

Who has abandoned email? Email is not the best tool for everything. That is true for every other messaging system.

"a secure-by-default environment"

That refers to smartphones? Funny.

The main problem – which also makes a comparison of systems difficult – is that we do not have standardized scales for system security (which is the limit to key security) and authentication quality. When talking about "security" most people refer to crypto details – which are hardly ever

the main security problem in real life.

Anonymous says:
August 14, 2014 at 2:44 am
Yes, it is, if by 'paranoia' you mean a legitimate contribution to the conversation.

While I am completely opposed to the NSA crap *and* I realize the illegitimacy of the "if you have nothing to hide" observation, it is also true that most of us are extremely unlikely to be harmed by the NSA et al. So unlikely, in my opinion, that it does not belong in a conversation about this considering the very real and likely harm resulting from other sorts of intrusion.

Michael says:
August 14, 2014 at 6:46 am
Our SafeSlinger project is working on easy to use secure key/fingerprint exchange and verification, even for remote users. Our protocol ensures a human to human out of band SAS comparison, that resists the habituation of users to click-through the verification process.

http://www.cylab.cmu.edu/safeslinger

Anonymous says:
August 14, 2014 at 7:19 am
I'm surprised by the gripes you left out:

– Key-signing parties often check government ID's but don't even check that you're able to receive email at the address you put in your key ID. The other partygoers accumulate signed keys on their keyring and then bulk upload them, perhaps emailing you a copy as a courtesy. This is exactly the wrong kind of endpoint binding for most PGP scenarios: either I want to know that I'm emailing the same person I met, so I need a business card, and don't need WoT at all. Or my friend gave me an email address and said "write this guy," and I want to know I'm really writing the guy my friend told me to write, so I need email addresses bound to keys. I don't care a bit about being able to sue the person later, or whether his work permit is fake. What is the point of the "government ID"? Lots of people have the same names as others. It's also poorly-validated: a fake ID good enough to pass a key-signing party is a low bar. If the goal is defense against spooks, they can just print a real ID with my name on it.

– PGP could allow you to sign a substring of the key ID, but it doesn't.

– The right policy for a key-signing party is something like "if someone introduces themself as oscar the grouch and then offers you a key with ID oscar the grouch, just sign it." That is WoT. You know me, I know this guy as Oscar, so he's Oscar. That's how searching for people on Facebook works, and it's great. A key-signing party with strangers is basically worthless. Actual key-signing parties do the total opposite of both of those rules: attempt to bring strangers together, and then if you are aware of a person's public reputation, exclude that awareness from leaking into any signing decisions.

– PGP key servers publish social graph data to the Internet where any spook can get it. For a real person at risk, social graph may be more sensitive than the email being encrypted. It enables the Paul Revere Metadata Attack, for example.

– PGP doesn't encrypt message headers, leaving Subjects and reply chains in the clear.

– PGP doesn't authenticate message headers. MUA's treat the unauthenticated To: field as the source of truth and display the PGP key ID in some ad-hoc fashion. If you accept my public key, I can sign a message, forge the unsigned To: field, and have a good chance of tricking you.

Consequently, if you are signing only, not doing encryption, DKIM-signed mail is more likely to frustrate an attack than PGP-signed mail.

Finally, why didn't you mention Pond? Do you think we need to keep awareness of it low until bugs are shaken out? It seems to address all your concerns, all mine, and some more besides.

Anonymous says:
August 14, 2014 at 8:30 am
But the moment you have to stop trusting Verisign for whatever reason (they are compromised, for example), you are screwed. This is a central point of failure. You can't sign your certs with multiple providers and expect them NOT to revoke your cert the moment they learn that Verisign was compromised, so it all goes down the drain quickly.

newlog says:
August 14, 2014 at 11:59 am
We're all harmed by the NSA et al. just with massive surveillance. There's no need for them to make you a specific objective to be harmed.
We use to forget all we lose when society, insted of directly us, is surveilled.

So yes, this belongs to the conversation.

Anonymous says:
August 14, 2014 at 12:50 pm
You said, "Pictures of my dachshunds". Where are they? We want pictures.

Anonymous says:
August 14, 2014 at 1:05 pm
Some reasonable or even good points, but there's still a couple things that are plain selfish and short-sighted, even contradictory. I'm going to leave the which is which as an exercise (because, you know, YOU are the professor, figure it out). I'm vaguely surprised you didn't mention _Why Johnny Can't Encrypt_.

There's also that many email implementors don't actually understand email, so that the general usership doesn't is no surprise. IMO the readiness to part with email stems from this: Inability to properly use it, rendering the message stream unreadable regardless of encryption. The major problem is that all the alternatives are needlessly limited in some way, like size of the user base, foreign ownership (US counts as such for 95% of the world population), requiring signing up ("with real name, or else!") and complete loss of control of your archive plus promises to have your every move sold to advertisers, and so on, and email is relatively easy to graft things on top on. The downside being that too many implementors fsck up and so we're left with unusable software. Even something as widely used as thunderbird+engimail cannot handle messages properly. EG see what happens to the on-the-wire format if you try to forward-and-encrypt a text+html/alternative. The folks at enigmail have been notified but refused to fix it. "Too hard." How can email be "too hard"? That's an admission of incompetence. Get better implementors. Beats trying to reinvent the wheel incompetently, any day.

Anonymous says:
August 14, 2014 at 1:34 pm
If you believe that trusted third party/CA's represent an improvement over direct key exchange you are naïve. Trusted CA's are only trusted until their not. They are constantly getting get hacked or forced via legal edict, or having somebody on the inside simply getting bribed – and then your keys are compromised. CA's are the weakest link in secure end to end cryptography by far. Other than that great article.

As an aside – Google is the worst privacy offender on this planet (so I opt out of anything google) why support their oauth and not facebook's? Maybe less anon posters if you support FB oauth.

Hugo Leisink says:
August 14, 2014 at 1:54 pm
What you do think about the cryptography in Hisser? http://www.hisser.eu/ It's a chat application for mobile devices. Unique in our solution is the usage of Diffie-Hellman. We send DH values along with every message, allowing Hisser clients to use a unique AES key for every message.

KTetch Dureek says:
August 14, 2014 at 5:12 pm
Some good points, and yes, I think PGP/GPG is probably on the way out.
However, until there's something to replace it with, what's the alternative?

And yes, I know the encryption standards are old, they're SHA1 in GnuPG, which has been depreciated since 2010, but *shrug* what's the alternative?

In many ways, PGP/GPG is the 'least bad' option. And so for now, It's one I'm sticking with.

Now, it may not be 'the best' for ultra-secret, 'burn before reading' communications, but for personal information, and stuff that shouldn't be publicly disclosed, it works for me.
It's also something I've been pushing local lawyers to implement, to communicate with clients, so it's a bit more trusted and secure.
And finally, with public statements, I sign them, in line, so that it's easy to check if my words were 'altered' – the signature would fail.

Anonymous says:
August 14, 2014 at 6:15 pm
The problem with centralized key storage – especially where the keys are stored by the same entity that holds the cyphertext – is that it's trivial for the government to simply require the company to hand over the keys.

I know a lot of people don't trust banks, but at least in the financial world, the laws are a lot stronger w.r.t. who gets access to your financial information and who can touch the contents of your safe deposit box.

Why not centralize key storage with your financial institution — so that way, even if the government compels mail providers to hand over the ciphertext, they'll have to ALSO subpoena a bank (swiss banks anyone?) to get the keys to decrypt it?

Hovhannes Tumanyan says:
August 14, 2014 at 7:48 pm
This comment has been removed by the author.

Hovhannes Tumanyan says:
August 14, 2014 at 7:50 pm
I agree with prior poster comments and would like to add a few points:

a. ProtonMail – no real security beyond a promise that they won't comply with subpoenas and won't look into E-mails. Looking under the hood, one can see that they even store private keys

b. PGP/PKI – not practical for an average person. Too much effort to install the tools, potentially switch to a desktop E-mail client, find peers and exchange keys etc. Bottom line – no way to get any meaningful adoption beyond the small circle of crypto professionals, enthusiasts and those

who handle highly sensitive data

c. Virtu – better than traditional PGP/PKI. Allows in-browser encryption. Has plugins for GMail and Yahoo. The big problem is that it focuses on encryption leaving out the key management. Virtu solves the key management problem through storage of key material on their servers – same kind of problem as with Proton Mail: need to trust a 3rd party

d. Yahoo and GMail offerings are, essentially nothing else but an attempt to re-pakage PGP to make installation more convenient. The underlying key management problem still remains

The bottom line is that all of these solutions are just marginally better from tools that existed for decades and that found no meaningful adoption due to the issues mentioned. I believe the fundamentally better solution is likely to come from NameCoin or other bitcoin-driven mechanisms and communities. NameCoin already provides a reliable and secure key distribution mechanism and allows E-mail addresses to be associated with public keys with no centralized infrastructure involved.

There is a Chrome extension, called SecureDolphin that enables simple, 3-step installation and key broadcast and enables encryption/decryption right in GMail and Yahoo. Check it out at SecureDolphin or in the Chrome Store

Anonymous says:
August 14, 2014 at 9:21 pm
You could always print them a business card and put them a safe deposit box.

Kevin Riggle says:
August 14, 2014 at 9:45 pm
Well, it's relatively easy to get a new cert issued by a different CA if Verisign gets compromised. But my point is that that's true of the web already, and normal users rarely have to care.

Yahoo Mail's certificate is, in fact, currently issued by Verisign, so encrypting mail with a client cert signed by Verisign doesn't expose me to anything I'm not exposed to already by trusting Yahoo and Verisign with my mail, and protects me against passive eavesdropping or active modification by anyone short of a nation-state actor.

PGP will still be around for people who are defending against nation-state attackers, but for the rest of us I think S/MIME provides much more real safety at much less cost.

Anonymous says:
August 15, 2014 at 1:25 pm
The whole point of the article was to argue that the failing implementations result from a flawed protocol. Simply observing that the implementations are broken (but the fundamental design of the protocol is not) will not fix anything, because you will not have understood WHY the implementations suck so much.

Anonymous says:
August 15, 2014 at 1:33 pm
Lavaboom? Not a chance. The way you present it makes it clear the key exchange is not completely independent from your own software cq. systems which means you are not truly zero-knowledge (or you may require users to TRUST that you are based on your word alone). Any zero-knowledge provider must rely on third-party tools or verifiable or auditable implementations (which is basically the same thing) to make sure the STORAGE is orthogonal to the ENCRYPTION. You can't have the same system or the same provider to manage or facilitate both.

nooblet says:
August 15, 2014 at 2:10 pm
so what would you advise for a novice to use?

Unknown says:
August 15, 2014 at 4:11 pm
You mean the Deep Web. The Darknet is a p2p network.

Unknown says:
August 15, 2014 at 4:26 pm
I smell NSA-back propaganda. Yahoo and Google both volunteer any petitioned information to the NSA; what's keeping them from volunteering encryption information, too?

Ralph Dratman says:
August 16, 2014 at 1:14 am
Anonymous, the problem with the NSA's type of universal eavesdropping, even for people with nothing to hide, is that individuals employed by NSA and its subsidiary organizations and contractors have the opportunity to abuse a system that is supposed to protect us. And, like Murphy's law, if a group of people can abuse power, inevitably some of them will actually do so. Then a system which is designed to do one thing can become an instrument of someone's personal problems, often with truly devastating consequences. Remember, Edward Snowden did not even work for the government. He worked for a company that was contracting to outsource some NSA work — yet supposedly had access to some of the nation's most sensitive secrets. Imagine if Snowden had been a malicious person instead of a patriotic one. Think of the damage he might have done to both people and institutions.

Anonymous says:
August 16, 2014 at 2:00 am
Off topic!
@Ralph Dratman: Holy shit! So, that's what Snowden was – patriotic. Is that the generally accepted (in the US) behavior for patriots or is it just your idea about patriotism? :))

Anonymous says:
August 16, 2014 at 9:30 am
This is a technical post; talk of NSA and whether Google can be trusted is trolling, pure and simple.

Anonymous says:
August 16, 2014 at 9:41 am
I smell a stupid troll.

Anonymous says:
August 16, 2014 at 6:35 pm
Matthew, all your arguments about the problems with PGP may be solid, but the real barrier to an alternative is the massive network effects locking people into email. Email is still the default messaging protocol on the internet, and practically every other service requires email (or SMS!) to register and to communicate with its users. While I am a fan of TextSecure and advocate for everyone to use it, it's not realistic to expect that everyone will switch off email and migrate to a new protocol. As long as almost everyone on the internet uses email, some encrypted email solution is required. And as long as most people are using webmail, some form of encrypted webmail solution is required.

I don't think Yahoo and Gmail breaking compatibility with existing PGP is a huge deal. For one thing, as you argue, the existing user base is very small. If Yahoo and Google succeed, legacy PGP users will be a niche group. Any user of Google/Yahoo confronting this case will probably be savvy enough to import compatible private keys to allow legacy PGP users to encrypt to them (since the other direction is not a problem). And it will force rapid adaptation by GnuPG et al.

As for the problem of key syncing, Google's End-To-End team says they are working on a good, private, usable solution. We'll see.

That being said, I think you could hone your argument to advocate for the following:

1. A long-term rearchitecting of the email protocol to remediate some or all of its defects (transparency of metadata in transit, e.g.) by baking in authentication and privacy.
2. Yahoo and Google developing a parallel protocol that would remediate some of the protocol defects and would transparently be used instead of email between any compatible federated servers (presumably including themselves). The fallbacks would be PGP + SMTP and plaintext SMTP.
3. Better UX design — I am sure they are working on it.
4. End-to-end encryption on by default in Yahoo Mail and Gmail. Of course, Google will never do this as it undermines their business model but they are clever enough, if they want, to solve the problem of indexing local encrypted data and using central APIs to provision ad content, Google Now, etc. Or they could offer a paid option to encrypt everything all the time.
5. Google and MS baking TextSecure into Android, Windows Phone and iOS as the default device-to-device messaging protocol.

Anonymous says:
August 16, 2014 at 7:02 pm
Oops, in #5 had originally intended to include Apple (hence the orphaned iOS reference) but thought better of the idea of Apple ever agreeing to such a thing since they want to lock users into their proprietary ecosystem.

Anonymous says:
August 16, 2014 at 11:31 pm
Another problem is PGP software tells the world the version you are using, which can identify what operating system you have, when you upgraded if messages monitored over time, and if you're using a vulnerable version of PGP as per above mentioned bugs.

Felix Fontein says:
August 17, 2014 at 11:18 am
"Forward secrecy is not an important feature for email. People want to read their data not throw it away."

I *strongly* disagree. Forward secrecy protects against two problems, and while one of them is indeed not so important, the other one is very important.

On the receiver's side, the needed information to decode the message can be stored together with the encrypted email (or unencrypted, if the user prefers). Then it is easily possible to read all emails (either directly if stored unencrypted, or with the user's private key if they are encrypted, as the other information needed — the forward secrecy data — is stored). Here, having forward secrecy is not needed at all and seems like additional expenses.

But consider transport of the mails. Every encrypted mail is seen in encrypted form by at least two servers (assuming sender and receiver use different servers) which are usually under control by neither sender nor receiver. Hence, these servers could store copies of the encrypted mails (or

forward them to someone else, like the NSA). As the email system is designed, you have essentially no possibility to ensure that this will not happen.

Now if the server has a long list of encrypted mails send to a user, they have to crack one public key in order to read *all* of these messages. But in case forward secrecy is used, they cannot read a single mail after cracking the public key. They have to crack a different ephemeral key for each message to be able to read it.

So forward secrecy *is* very important for emails as well. Also, it allows the sender and receiver to destroy all evidence on the contents of this email by deleting the ephemeral key from their hard disks. If they are later forced to hand their hard disks over, nobody can simply extract the ephemeral key for these messages. This is not possible when not having forward secrecy: there you can decide only between giving up access to all your encrypted data (by deleting the private key), or by keeping it and allowing everyone who gets his hands on your private key to read everything that was ever encrypted with this key.

Dave says:
August 17, 2014 at 4:06 pm
The crypto-museum problem (nice label ☺ mostly arises from GPG, not specifically any general OpenPGP implementation. I still have to maintain support for these antique ciphers from the 1990s solely because GPG refuses to update to any current cipher. Ian Grigg has a good comment on this on the cryptography mailing list, "The people that love this algorithm agility idea love to *add* algorithms but have no process to take them away". The name for them is "zombie algorithms", undead algorithms that just refuse to go away no matter how many times you try and kill them.

Anonymous says:
August 17, 2014 at 4:23 pm
Protonmail was vulnerable to a simple javascript injection attack, it doesn't belong in any serious crypto conversation.

Anonymous says:
August 18, 2014 at 9:22 am
How can we trust minilock and cryptocat as long as idiot savant and crypto noob Nadim Kobeissi is in charge of them?

Anonymous says:
August 18, 2014 at 11:08 pm
Because the source is open so you can make yourself a useful human being by reviewing it rather than wasting everyone's time by being a jerk in internet comment threads.

Anonymous says:
August 19, 2014 at 8:54 am
Remind me how that worked with OpenSSL

Bernhard says:
August 19, 2014 at 12:54 pm
Good article! IMO the worst and most important point of all is "the terrible mail client implementations". PGP was/is just used by a few experts. And because of that the software usability isn't really improved. Software gets usable not before the masses catch up with it.

Anonymous says:
August 19, 2014 at 8:16 pm

OpenSSL is a massive dinosaur with a huge codebase. Minilock isn't. And unlike some random internet snide-ass commenters, other people have actually looked at it.

Anonymous says:
August 19, 2014 at 9:24 pm
Central points of failure are not only, or even primarily, in danger from nation-state attackers. The entire idea that my business secrets or personal secrets should be only as safe as some third party makes them, one over whom I have no control and no visibility as to whether they are actually compromised, is a non-starter for real security. It is a "just trust us" model. I don't want to, nor should I have to, trust Verisign or anyone other than the person I am actually corresponding with, in order to have a protected conversation with that person.

Brad Jensen says:
August 20, 2014 at 6:25 am
What we really want is not the Perfect Cryptographic Solution, but rather the reasonable expectation that we can send emails to a recipient without them being in clear text and available to government and criminal snoops along the way.

We don't want a separate system for sending some emails, we want all emails to be secure. Fortunately there is a much simpler solution than having billions of users get public and private keys and dance Gangnam Style every time they want to send a message.

Instead, add ssl transmission to all pop, IMAP, and SMTP servers. Then add the same capability to all email clients. It really isn't big job if you give up the idea of authenticating the sender and receiver – which is basically impossible to do at the present time anyways.

Servers can continue to support insecure clients and insecure servers forever.

An email client sending an email attaches a private hash in an extra header that includes the from email address. If the email is sent through a web interface, the email service provider adds the header.

The smtp server than refuses to pass along the header to any other server that it connects to insecurely, and the same with the POP3 or IMAP server at the other end.

You can't spoof the hash because it is private to the original sender. The client at the other end sends back the same hash when that email user sends an email back to the original sender. So the sender is informed whether the email was securely transferred during the roundtrip. To establish the secure roundtrip channel initially, you could use a special poke message.

Each sender authenticates themselves, and the servers in the middle can pass along the header but not spoof it. Since most of us send our emails through the same server or service all the time for each email address, all we have to trust is our local server. We basically have to trust the local server not to pass along the security header in insecure conversations.

The sender can change their hash periodically or on a random basis.

I suggest Google set up the specifications for this and invite others to support it, so it doesn't get bogged down in committees and standards organizations for years.

This would be simple, cheap, and pretty universal in a very short amount of time.

Anonymous says:
August 21, 2014 at 4:13 pm

TROLL!!!!
…
TROLL!!!!!!
…

TROLL!!!!

Don't waste your time reading this ridiculous article!

Anonymous says:
August 22, 2014 at 11:42 pm
Kevin,
I get what you mean about a trusted CA model being more transparent for most end users, but I'd suggest that's a non-human model that most have gotten used to and pretty much devalues the whole concept of real world trust. There are few entities, if any, that i'd ever give absolute control over determining who I should trust. The PGP WoT was designed to model human behavior not machine automation or behavior. As people, we don't give blanket trust just because that would make things easier. We trust who we trust for real reasons. If i haven't met someone before, I have nothing to base my trust on. The WoT had the concept of trusted introducers and multiple levels of depth to support more of a corporate trust model, where the company acts as a type of CA for issuing, signing, and establishing trust automatically/synthetically for the purpose of doing business. Anyhow, the main point i wanted to make is that the X.509 CA model is bullshit, synthetic trust meant to try to solve a human problem with a machine. Computers are not capable of this concept. Just because the X.509 CA trust model is easiest for machines to deal with, it doesn't mean that this solves any real human problem.

Sara Soratia says:
August 23, 2014 at 8:44 am
Hi,

In my company for security purposes PGP Encryption is mandatory. Allthough this puts down the laptop (boot time is around 40 minutes for a 2Ghz Core2Duo) not to mention how it works in general. In order to solve this I decided to deploy the windows on a SSD. After installing everything I had to install I was amazed with the performance of the pc with cloudwedge. Ok, this is fantastic. thanks

Anonymous says:
August 25, 2014 at 5:33 pm
Is there some reason we need to have a single, monolithic level of security for all email users? Most of us aren't sending information any more sensitive than PII (SSNs, credit card info, the usual stuff identity thieves go after). Whatever security would protect those people, let's call it Level 1. This is what everyone should have at a minimum. Once you start sending around sensitive info about corporate or governmental malfeasance (e.g. up to and including Snowden-level stuff), you need more involved security. Let's call that Level 2. Beyond that is, like, secret government and military communications. Let's call that Level 3.

Level 1 could be as simple as having email clients contain a built-in, automatic key exchange mechanism that's handled through X- headers. Your client automatically creates and shares keys with each correspondent that you send an email to. After two emails back and forth, now you've exchanged keys and all further comms between you two are encrypted and signed. If you receive any email that isn't properly signed, it goes into an "Untrusted" folder in your client, for you to peruse later. This way, if you get spam, they won't ever* be signed, and it's easy to filter them out.

Yes, it's possible for a correspondent's computer to become compromised, and then a virus sends you spam as that user, and it goes into your "Trusted" folder because it's signed. There's no way to defend against this, and you just have to use your brain to notice when you're getting spam or weird attachments or suchlike. But with proper filtering (or a mail client that's not stupid), it's easy to have mail from each correspondent sequestered in its own box, just like if you were communicating with people on Facebook or Hangouts.

This can be combined with a unique-address system, where instead of just having one email address, you have one email address per correspondent. Gmail already does this via the "foo+bar@gmail.com" system; anyone with their own email domain can do this too. Now, whenever you give someone your email address, you just give them a unique address that no one else has. That way, if you start getting spam to that address, you can just filter that one address out and give them another one (or not). And of course you can combine this with typical Bayesian filtering to avoid even more spam.

Level 2 and above would require more technical know-how and involvement, since the danger of discovery is so much greater. For most of us, Level 1 is plenty.

Bart says:
August 25, 2014 at 6:38 pm
Could you address arguments provided in https://pthree.org/2014/08/18/whats-the-matter-with-pgp/ ?

Anonymous says:
August 25, 2014 at 9:12 pm
With the multitude of laws out there, with surely more and more laws on top of those to come, how do you know that you have nothing to hide? That is the same reason you must refuse to answer any question from the Boarder Patrol beyond what is your citizenship. All further questions are for one reason, to trap you in your words. You definitely always have something to hide.

Anonymous says:
August 26, 2014 at 11:47 am
I'm even more surprised there's no mention of Garfinkle & Miller's Johnny 2 paper, which takes these topics on head-on.

Anonymous says:
August 26, 2014 at 2:45 pm
You can almost fix one issue with PGP if you use an "ephemeral PGP key" application which utilizes PGP only to sign/encrypt messages: Before encryption, a future reply public PGP key is attached to the message and the current secret PGP key is deleted after the encryption. The same application can be used to reply with a newly created reply key. Not the same as with Diffie-Hellman key exchange but as close as it can be with a passive air gapped PC.

Anonymous says:
August 27, 2014 at 6:12 am
Well, it takes one to smell one!… So No… His was a fair comment.
Disinformation, posturing, sabotage, deception and manipulation are part of any security equation here!
You can't meaningfully discuss security without considering the various agendas and potential forms of attack.

Anonymous says:
August 27, 2014 at 6:49 pm

Really? For One solved bug? Or do you have some "serious crypto" arguments?

Rafael says:
August 28, 2014 at 10:04 am
I agree with many of your concerns, but use Apple software or any other propietary software/OS/Solution just throws away the security, because one is sitting on a black box with backdoors.

Uriel Fanelli says:
August 28, 2014 at 6:13 pm
Basically you are saying: "I can't find a single click icon to do all the job on my mac, then it sucks". You put lot of useless words there, but actually this is the meaning.

I hope macintosh guys will make themselves a reason: text exists. Some text is not something you can remember.

And it still exists.

Yes, I know, the average mac user can't realize how something difficult can exist when on apple "everything is easy". Well, the point is, life is not supposed to be easy.

Neither security.

Anonymous says:
August 29, 2014 at 6:05 am
No key shoyld be uploaded to any server because server can record your IP address and then the key is connected to you.

Beside it, any new product should be checked, if you check background of silent circle, you will see that they are military intelligence and you should not trust them, they are spies 100%:
https://blog.predicsasa.com/computers/silent-circle-is-nsa-shit/

Spud says:
September 1, 2014 at 3:32 am
Nice list, but you forgot repudiation (a la OTR). Easy to be cynical about NSA^H^Hgoogle pushing non-repudiated messaging on everyone.

Anonymous says:
September 2, 2014 at 10:03 am
Traceback at http://www.z80.eu/blog/index.php?entry=entry140901-180000

I totally disagree about this unreflected blog entry.
Sure, PGP should be improved in terms of usability.
No, there is no real alternative if you want to get data integrity.
I am NOT a friend of central aka cloud solutions (like you?).

Anonymous says:
September 4, 2014 at 3:06 pm
Maybe https://bitmessage.org (based on bitcoin protocol) is an alternative.

Anonymous says:
September 8, 2014 at 7:04 pm
I haven't seen any mention of RMail encryption in these comments? As I understand with RMail, the original message and attachment are encrypted to PDF before sending (Keys included), and remain encrypted for the whole journey. Even when they land they remain encrypted until the

password is use to open.

For me any service that relies on another server promising not to take a look, hand over to big brother or hide behind country laws cannot be trusted. The fact that google and yahoo are even mentioning encryption makes my toes curl with hypocrisy.
RMail http://www.mailsoftsolutions.co.uk

Anonymous says:
September 11, 2014 at 10:04 pm
To summarize this article I must say: There is no magic button to make things happen.
We MUST use what we have and say THANK YOU for it. If you want something better, the way is: read, study, research and do it better.

We are living in an accommodated society where everything must be easy and fast with no real values.

What is your contribution?

Comments are closed.

Menu