▲ Gpg.fail (gpg.fail)

450 points by todsacerdoti 4 months ago | hide | past | favorite | 357 comments

▲ oefrha 4 months ago | next [–]

Okay, since there's so much stuff to digest here and apparently there are issues designated as wontfix by GnuPG maintainers, can someone more in the loop tell us whether using gpg signatures on git commits/tags is vulnerable? And is there any better alternative going forward? Like is signing with SSH keys considered more secure now? I certainly want to get rid of gpg from my life if I can, but I also need to make sure commits/tags bearing my name actually come from me.

▲ tptacek 4 months ago | parent | next [–]

One of those WONTFIX's is on an insane vulnerability: you can bitflip known plaintext in a PGP message to switch it into handling compression, allowing attackers to instruct GnuPG packet processing to look back to arbitrary positions in the message, all while suppressing the authentication failure message. GPG's position was: they print, in those circumstances, *an error of some sort*, and that's enough. It's an attack that reveals plaintext bytes!

▲ upofadown 4 months ago | root | parent | next [–]

Are you referring to "Encrypted message malleability checks are incorrectly enforced causing plaintext recovery attacks"?

Seems like a legitimate difference of opinion. The researcher wants a message with an invalid format to return an integrity failure message. Presumably the GnuPGP project thinks that would be better handled by some sort of bad format error.

The exploit here is a variation on the age old idea of tricking a PGP user into decrypting an encrypted message and then sending the result to the attacker. The novelty here is the idea of making the encrypted message look like a PGP key (identity) and then asking the victim to decrypt the fake key, sign it and then upload it to a keyserver.

Modifying a PGP message file will break the normal PGP authentication[1] (that was not acknowledged in the attack description). So here is the exploit:

* The victim receives a unauthenticated/anonymous (unsigned or with a broken signature) message from the attacker. The message looks like a public key.

* Somehow (perhaps in another anonymous message) the attacker claims they are someone the victim knows and asks them to decrypt, sign and upload the signed public key to a keyserver.

* They see nothing wrong with any of this and actually do what the attacker wants ignoring the error message about the bad message format.

So this attack is also quite unlikely. Possibly that affected the decision of the GnuPG project to not change behaviour in this case, particularly when such a change could possibly introduce other vulnerabilities.

[1] https://articles.59.ca/doku.php?id=pgpfan:pgpauth

Added: Wait. How would the victim import the bogus PGP key into GPG so they could sign it? There would normally be a preexisting key for that user so the bogus key would for sure fail to import. It would probably fail anyway. It will be interesting to see what the GnuPG project said about this in their response.

▲ tptacek 4 months ago | root | parent | next [–]

In the course of this attack, just in terms of what happens in the mechanics of the actual protocol, irrespective of the scenario in which these capabilities are abused, the attacker:

(1) Rewrites the ciphertext of a PGP message

(2) Introducing an entire new PGP packet

(3) That flips GPG into DEFLATE compression handling

(4) And then reroutes the handling of the subsequent real message

(5) Into something parsed as a plaintext comment

This happens without a security message, but rather just (apparently) a zlib error.

In the scenario presented at CCC, they used the keyserver example to demonstrate plaintext exfiltration. I kind of don't care. It's what's happening under the hood that's batshit; the "difference of opinion" is that the GnuPG maintainers (and, I guess, you) think this is an acceptable end state for an encryption tool.

▲ akulbe 4 months ago | root | parent | prev | next [–]

Is there a better alternative to GPG?

▲ tptacek 4 months ago | root | parent | next [–]

Everything is better than PGP (not just GPG --- all PGP implementations).

The problem with PGP is that it's a Swiss Army Knife. It does too many things. The scissors on a Swiss Army Knife are useful in a pinch if you don't have real scissors, but tailors use real scissors.

Whatever it is you're trying to do with encryption, you should use the real tool designed for that task. Different tasks want altogether different cryptosystems with different tradeoffs. There's no one perfect multitasking tool.

When you look at the problem that way, surprisingly few real-world problems ask for "encrypt a file". People need backup, but backup demands backup cryptosystems, which do much more than just encrypt individual files. People need messaging, but messaging is *wildly* more complicated than file encryption. And of course people want packet signatures, ironically PGP's most mainstream usage, ironic because it relies on only a tiny fraction of PGP's functionality *and still somehow doesn't work*.

All that is before you get to the absolutely deranged 1990s design of PGP, which is a complex state machine that switches between different modes of operation based on attacker-controlled records (which are mostly invisible to users). Nothing modern looks like PGP, because PGP's underlying design predates modern cryptography. It survives only because nerds have a parasocial relationship with it.

▲ palata 4 months ago | root | parent | next [–]

> It survives only because nerds have a parasocial relationship with it.

I really would like to replace PGP with the "better" tool, but:

* Using my Yubikey for signing (e.g. for git) has a better UX with PGP instead of SSH

* I have to use PGP to sign packages I send to Maven

Maybe I am a nerd emotionally attached to PGP, but after a year signing with SSH, I went back to PGP and it was so much better...

▲ computerfriend 4 months ago | root | parent | next [–]

> better UX with PGP instead of SSH

This might be true of comparing GPG to SSH-via-PIV, but there's a better way with far superior UX: derive an SSH key from a FIDO2 slot on the YubiKey.

▲ palata 4 months ago | root | parent | next [–]

I do it with FIDO2. It's inconvenient when having multiple Yubikeys (I always end up adding the entry manually with ssh-agent), and I have to touch the Yubikey everytime it signs. That makes it very annoying when rebasing a few tens of commits, for instance.

With GPG it just works.

▲ ahlCVA 4 months ago | root | parent | next [–]

For what it's worth: You can set no-touch-required on a key (it's a generation-time option though).

▲ palata 4 months ago | root | parent | next [–]

Sure, but then it is set to no-touch for every FIDO2 interaction I have. I don't want to touch for signing, but I want to touch when using it as a passkey, for instance.

▲ ahlCVA 4 months ago | root | parent | next [–]

This is a per-credential setting, so you can have your SSH signing key be a no-touch key and still use touch confirmation for everything else.

(see "uv" option here https://fidoalliance.org/specs/fido-v2.0-ps-20190130/fido-cl... - the -sk key types in SSH are just a clever way of abusing the FIDO protocol to create a signing primitive)

▲ palata 4 months ago | root | parent | next [–]

Oh, I need to check this! Thanks!

▲ Avamander 4 months ago | root | parent | prev | next [–]

Use the PIV applet for SSH and signing Git commits instead? Git supports S/MIME and SSH can use keys over PKCS#11 basically out-of-box on OSs that don't ship gpg-agent (that just interferes with SmartCard usage in general).

▲ johnisgood 4 months ago | root | parent | prev | next [–]

Now can you give us a list of all the features of PGP and a tool that does one specific thing really well?

▲ some_furry 4 months ago | root | parent | next [–]

https://soatok.blog/2024/11/15/what-to-use-instead-of-pgp/

I wrote this to answer this exact question last year.

▲ palata 4 months ago | root | parent | next [–]

> The only downside to Sigstore is it hasn't been widely adopted yet.

Which, from where I stand, means that PGP is the only viable solution because I don't have a choice. I can't replace PGP with Sigstore when publishing to Maven. It's nice to tell me I'm dumb because I use PGP, but really it's not my choice.

> Use SSH Signatures, not PGP signatures.

Here I guess it's just me being dumb on my own. Using SSH signatures with my Yubikeys (FIDO2) is very inconvenient. Using PGP signatures with my Yubikeys literally *just works*.

> Encrypted Email: Don’t encrypt email.

I like this one, I keep seeing it. Sounds like Apple's developer support: if I need to do something and ask for help, the answer is often: "Don't do it. We suggest you only use the stuff that just works and be happy about it".

Sometimes I have to use emails, and cryptographers say "in that case just send everything in plaintext because eventually some of your emails will be sent in plaintext anyway". Isn't it like saying "no need to use Signal, eventually the phone of one of your contacts will be compromised anyway"?

▲ Natanael_L 4 months ago | root | parent | next [–]

The fact that every email encryption integration exports secure context messages into insecure contexts when decrypting (which is how encrypted messages end up cited in plaintext) means email can't be secured.

This is true both for GPG and S/MIME

Email encryption self-compromises itself in a way Signal doesn't

▲ some_furry 4 months ago | root | parent | prev | next [–]

> Which, from where I stand, means that PGP is the only viable solution because I don't have a choice.

You don't have a choice *today*. You could have a choice tomorrow if enough people demanded it.

Don't let PGP's convenience (in this context) pacify you from making a better world possible.

▲ palata 4 months ago | root | parent | next [–]

I agree with that. But I feel like I have been reading for years that there is really no reason to use PGP, and I have tried for years to use alternatives, but the fact remains that I still need to use PGP, either because it is mandatory or because in some cases the alternatives are not practical.

To me, there will be no reason to use PGP the day I find practical alternatives for the remaining use-cases I have. And I feel like signing git commits is not a weird use-case...

▲ tptacek 4 months ago | root | parent | next [–]

Does the GnuPG project sign its git commits with PGP?

▲ KooBaa 4 months ago | root | parent | next [–]

Of course it does, and all released software and tarballs as well.

▲ palata 4 months ago | root | parent | prev | next [–]

Not sure what you are trying to say

▲ xeonmc 4 months ago | root | parent | prev | next [–]

offtopic question:

as a recent dabbling reader of introductory popsci content in cryptography, I've been wondering about what are the different segmentation of expert roles in the field?

e.g. in Filippo's blogpost about Age he clarified that he's not a cryptographer but rather a cryptography engineer, is that also what your role is, what are the concrete divisions of labor, and what other related but separate positions exists in the overall landscape?

where is the cutoff point of "don't roll your own crypto" in the different levels of expertise?

▲ Natanael_L 4 months ago | root | parent | next [–]

There's no clear segmentation. There's symmetric and asymmetric primitives (and stuff that doesn't fit into these like ZKP), algorithms, protocols, research in many different types of attacks against each of these, research in design and defenses, and plenty of people will cover completely different subsets.

"don't" roll your own cover everything from "don't design your own primitive" to "don't make your own encryption algorithm/mode" to "don't make your own encryption protocol", to "don't reimplement an existing version of any of the above and just use an encryption library"

(and it's mostly "don't deploy your own", if you want to experiment that's fine)

▲ xeonmc 4 months ago | root | parent | next [–]

I wonder if there is a concrete point at which it turn into “this is common sense security that even you should know about” like not conflating hashing and encryption, or “you should just have someone else do do security for you”? I guess at larger entities you have a CISO role but what about in smaller, scrappy endeavours, how does one know where one is at the limit of their due-commonsense and hand it off?

▲ tptacek 4 months ago | root | parent | next [–]

Most practitioners in security --- from information security to compliance to systems security to software security to red-teaming --- have very little competence with cryptography. Cryptography is hyperspecialized. It is not part of the toolkit of any ordinary professional.

(That's nothing to do with how hard cryptography is, just with how little demand there is for serious cryptography engineering, especially compared with the population of people who have done serious academic study of it.)

▲ tptacek 4 months ago | root | parent | prev | next [–]

There isn't one, but the modal professional cryptography engineer probably has a graduate degree in cryptography.

▲ some_furry 4 months ago | root | parent | prev | next [–]

My job title is in the Security Engineer family.

I do not have a Ph.D in Cryptography (not even an honorary one), so I do not call myself a Cryptographer. (Though I sometimes use "Cryptografur" in informal contexts for the sake of the pun.)

▲ pseudohadamard 4 months ago | root | parent | next [–]

What you actually want doing crypto is a security engineer, not a cryptographer. To quote Shamir's Law, "cryptography is bypassed, not attacked". No-one ever attacks the crypto, they attack the way it's used, so you need an experienced cryptoplumber to set it up correctly, not a cryptographer who will design a mathematically elegant whatsit and announce "there, solved!".

Ideally, this person will also design the system that uses the crypto, because no matter how skilled the people on a standards committee might be their product will always be, at best, a baroque nightmare with near-infinite attack surface, at worst an unusable pile of crap. IPsec vs. Wireguard is a prime example, but there are many others.

▲ xeonmc 4 months ago | root | parent | prev | next [–]

Interesting. In a general sense, where does it fall on the xkcd#435 scale?

▲ johnisgood 4 months ago | root | parent | prev | next [–]

You did not ask me, but you should do your due diligence because there are way too many armchair cryptographers around here.

▲ miki123211 4 months ago | root | parent | prev | next [–]

This is exactly that, in more detail than you could possibly ever ask for:

https://soatok.blog/2024/11/15/what-to-use-instead-of-pgp/

▲ akerl_ 4 months ago | root | parent | prev | next [–]

https://www.latacora.com/blog/2019/07/16/the-pgp-problem/#th...

▲ jhgb 4 months ago | root | parent | next [–]

> Use Signal. Or Wire, or WhatsApp, or some other Signal-protocol-based secure messenger.

That's a "great" idea considering the recent legal developments in the EU, which OpenPGP, as bad as it is, doesn't suffer from. It would be great if the author updated his advice into something more future-proof.

▲ akerl_ 4 months ago | root | parent | next [–]

There's no future-proof suggestion that's immune to the government declaring it a crime.

If you want a suggestion for secure messaging, it's Signal/WhatsApp. If you want to LARP at security with a handful of other folks, GPG is a fine way to do that.

▲ goldsteinq 4 months ago | root | parent | next [–]

> If you want a suggestion for secure messaging, it's Signal/WhatsApp. If you want to LARP at security with a handful of other folks, GPG is a fine way to do that.

I want secure messaging, not encrypted SMS. I want my messages to sync properly between arbitrary number of devices. I want my messaging history to not be lost when I lose a device. I want not losing my messaging history to not be a paid feature. I want to not depend on a shady crypto company to send a message.

▲ tptacek 4 months ago | root | parent | next [–]

I seriously don't care what messenger you use, as long as it isn't email, which can't be made secure. Pick something open source. It'll be less secure than Signal, but way more secure than email.

▲ Natanael_L 4 months ago | root | parent | prev | next [–]

Then your next best bet is Matrix.org. Not to the same security standard as Signal, but if you don't have a specific threat against you then it's fine.

▲ goldsteinq 4 months ago | root | parent | next [–]

Pros of Matrix: it actually has a consistent history (in theory); no vendor lock-in. Cons of Matrix: encryption breaks constantly. Right now I'm stuck in a fun loop of endlessly changing recovery keys: https://github.com/element-hq/element-web/issues/31392

▲ Arathorn 4 months ago | root | parent | next [–]

bleurgh. that issue is very actively under investigation (modulo xmas). please can you submit debug logs from Element Web referencing that issue.

▲ goldsteinq 4 months ago | root | parent | next [–]

I'm facing it on Element Desktop, but I'll try to reproduce it on Element Web. I've tried to submit logs from Element Desktop, but it says that `/rageshake` (which I was told to do) is not a command. I'm happy to help with debugging this, but I'm not sure how to submit logs from Desktop.

Something like this happens basically every time I try to use Matrix though. Messages are not decrypting, or not being delivered, or devices can't be authenticated for some cryptic reason. The reason I even tried to use Element Desktop is because my nheko is seemingly now incapable of sending direct messages (the recepient just gets infinite "waiting for message").

▲ Arathorn 4 months ago | root | parent | next [–]

Weird. Encryption these days (in Element Web/Desktop and Element X at least) should be pretty robust - although this whole identity reset thing is a known bug on Element Web/Desktop. You can submit debug logs from Settings: Help & About: Submit Debug Logs, and hopefully that might give a hint on what's going wrong.

▲ goldsteinq 4 months ago | root | parent | next [–]

No "Submit Debug Logs" there, as far as I can see. Do I need to be on matrix.org homeserver for this to work or something?

https://photos.goldstein.lol/share/OIgowBN4Wmi4zlm8DmDP0s8jH...

▲ Arathorn 4 months ago | root | parent | next [–]

looks like whoever's run that Element has disabled debug log reporting. not sure i can do much to help here :/

▲ some_furry 4 months ago | root | parent | prev | next [–]

> I want secure messaging, not encrypted SMS.

I send long messages via Signal, typed on a desktop computer, all the time. (In fact, I almost exclusively use Signal through my desktop app.)

You don't have to use it like "encrypted SMS"! You're free.

> I want my messages to sync properly between arbitrary number of devices. I want my messaging history to not be lost when I lose a device.

OK. https://signal.org/blog/a-synchronized-start-for-linked-devi...

> I want not losing my messaging history to not be a paid feature.

I genuinely don't understand what you mean here. From https://signal.org/blog/introducing-secure-backups/

"If you do decide to opt in to secure backups, you'll be able to securely back up all of your text messages and the last 45 days' worth of media for free."

If you have a metric fuckton of messages, that does cost money, sure, but as they say:

"If you want to back up your media history beyond 45 days, as well as your message history, we also offer a paid subscription plan for US$1.99 per month."

"This is the first time we've offered a paid feature. The reason we're doing this is simple: media requires a lot of storage, and storing and transferring large amounts of data is expensive. As a nonprofit that refuses to collect or sell your data, Signal needs to cover those costs differently than other tech organizations that offer similar products but support themselves by selling ads and monetizing data."

If you want Signal to host the encrypted storage, that costs money. If you don't want to pay Signal money, they provide 45 days of backup for free.

If you want to self-host your own backups (at your own cost), that's easy to do.

https://imgur.com/a/ElfaIee

You can literally set up SyncThing to stream your on-device backups to your NAS, cloud storage, or whatever.

> I want to not depend on a shady crypto company to send a message.

Shady crypto company?

Are you referring to MobileCoin? That feature isn't in the pipeline for sending messages.

I checked! https://soatok.blog/2025/02/18/reviewing-the-cryptography-us...

▲ goldsteinq 4 months ago | root | parent | next [–]

> You don't have to use it like "encrypted SMS"! You're free.

Using it as something more than encrypted SMS requires persistent message history between devices.

> metric fuckton of messages

“More than 45 days” is a metric fuckton? Seriously?

> If you want Signal to host the encrypted storage, that costs money. If you don't want to pay Signal money, they provide 45 days of backup for free.

I don’t want Signal to store my messages. I want Signal to not lock in my messages on their servers, so I can sync them between my devices and back them up into my own backups.

> If you want to self-host your own backups (at your own cost), that's easy to do.

Except there’s no way to move it between platforms. I have more than one device.

> Are you referring to MobileCoin? That feature isn't in the pipeline for sending messages.

I don’t want shady crypto company to hold my data hostage, and there’s no way to store it on my hardware and then move it between platforms. That’s my problem with signal.

> A Synchronized Start for Linked Devices

It only properly transfers 45 days. You can’t have more than one phone. Phones are special “primary devices” and AFAIK you can’t restore your messages if you lose your phone even if you have logged-in Signal Desktop.

▲ some_furry 4 months ago | root | parent | next [–]

I literally included a screenshot that shows you can setup backups in a directory on your device and then use your own backup solution.

Signal is not holding you hostage.

▲ goldsteinq 4 months ago | root | parent | next [–]

Yes, if your only device is a single Android phone you can do that. You can’t, however, use that backup to populate your message history on other platforms.

I’ve already lost message history consistency because one of my devices was offline for too long. The messages are there on my other device, but Signal refuses to let me copy my data from one of my devices to another. Signal is, quite literally, worse at syncing message history than IRC — at least with IRC I can set up a bouncer and have a consistent view of history on all of my devices, but there’re no Signal bouncers.

▲ tptacek 4 months ago | root | parent | next [–]

Look, if defending "message history consistency" is a reason you're choosing some other secure messenger rather than Signal, then I don't think this argument is very productive; use some other secure messenger then. But if "message history consistency" is a reason you're endorsing encrypted email over Signal, you're committing malpractice.

The point is that whatever secure messenger you use, it must plausibly be secure. Email cannot plausibly be made secure. Whatever other benefits you might get from using it --- federation, open source, UX improvements, universality --- come at the cost of grave security flaws.

Most people who use encrypted email are doing so in part because it does not matter if any of their messages are decrypted. They simply aren't interesting or valuable. But in endorsing a secure messenger of any sort, you're influencing the decisions of people whose messages are extremely sensitive, even life-or-death sensitive. For those people, federation or cross-platform support *can't* trump security, and as practitioners we are obligated to be clear about that.

▲ goldsteinq 4 months ago | root | parent | next [–]

I'm definitely not "commiting malpractice" on account of not being a security practicioner. I'm talking from a perspective of a user.

It's important to me — as a user — that a communication tool doesn't lose my data, and Signal already did. Actual practicioners keep recommending Signal and sure, I believe that in a weird scenario where my encryption keys are somehow compromised without also compromising my local message history, Signal's double-ratchet will do wonders — but it doesn't actually work as a serious communication tool.

It's also kinda curious that while the "email cannot be made secure" mantra is constantly repeated online, basically every organization that needs secure communication uses email. Openwall are certainly practicioners, and they use PGP-over-email: are they commiting malpractice?

▲ joshuamorton 4 months ago | root | parent | next [–]

Very few organizations need security from state level or similar threats *and* the infrastructure provider. Most organizations that want secure email don't use any kind of e2ee at all, they just trust Google or Microsoft or whomever.

The few jobs that *actually* care about this stuff, like journalists, do use signal.

Openwall doesn't get security via pgp, it gets a spam filter.

▲ akerl_ 4 months ago | root | parent | prev | next [–]

> but it doesn't actually work as a serious communication tool.

Say more. Plenty of people use Signal as a serious communication tool.

> Openwall are certainly practicioners, and they use PGP-over-email: are they commiting malpractice?

They, and other communities that use GPG-encrypted emails are LARPing, and it’s only fine because their emails don’t actually matter enough for anybody to care about compromising them.

It’s not malpractice to LARP: plenty of people love getting out their physical or digital toys and playing pretend. But if you’re telling other people that your foam shield can protect them from real threats, you are lying.

▲ goldsteinq 4 months ago | root | parent | next [–]

> Say more. Plenty of people use Signal as a serious communication tool.

I did say more already. Maybe you believe in serious communication tools that can’t synchronize searchable history between devices, but I don’t.

> They, and other communities that use GPG-encrypted emails are LARPing, and it’s only fine because their emails don’t actually matter enough for anybody to care about compromising them.

Are we talking about the same Openwall? Are you aware what Openwall’s oss-security mailing list is? Please, do elaborate how nobody cares about getting access to an unlimited stream of zerodays for basically every Unix-like system.

▲ tptacek 4 months ago | root | parent | next [–]

At this point you're just repeating the argument you made upthread without responding to any of its rebuttals. That's fine; I too am comfortable with the arguments on this thread as they stand. Let's save each other some time and call it here.

▲ akerl_ 4 months ago | root | parent | prev | next [–]

I’m very familiar with oss-security, a public mailing list that doesn’t really have anything to do with GPG-encrypted emails. Encrypting emails to a public

mailing list, with GPG or otherwise, wouldn't really make sense.

▲ goldsteinq 4 months ago | root | parent | next [–]

Okay, sorry, not oss-security mailing list, oss-security _distros_ mailing list.

https://oss-security.openwall.org/wiki/mailing-lists/distros

> Only use these lists to report security issues that are not yet public

> To report a non-public medium or high severity 2) security issue to one of these lists, send e-mail to distros [at] vs [dot] openwall [dot] org or linux [dash] distros [at] vs [dot] openwall [dot] org (choose one of these lists depending on who you want to inform), preferably PGP-encrypted to the key below.

▲ akerl_ 4 months ago | root | parent | next [–]

Yes, that would be an example of LARPing security. The obviously indicat

or is that encrypting your message is entirely optional, per their own instructions. The less obvious bit is that even if you encrypt your message, anyone without GPG configured who replies has stripped any attempt at encryption from the contents.

▲ tptacek 4 months ago | root | parent | prev | next [–]

Yes.

▲ jhgb 4 months ago | root | parent | prev | next [–]

Nobody decided that it's a crime, and it's unlikely to happen. Question is, what do you do with mandatory snooping of centralized proprietary services that renders them functionally useless aside from "just live with it". I was hoping for actual advice rather than a snarky non-response, yet here we are.

▲ Fnoord 4 months ago | root | parent | next [–]

> Nobody decided that it's a crime, and it's unlikely to happen.

Which jurisdiction are you on about? [1] Pick your poison.

For example, UK has a law forcing suspects to cooperate. This law has been used to convict suspects who weren't cooperating.

NL does not, but police can use force to have a suspect unlock a device using finger or face.

[1] https://en.wikipedia.org/wiki/Key_disclosure_law

▲ closewith 4 months ago | root | parent | prev | next [–]

You're asking for a technical solution to a political problem.

The answer is not to live with it, but become politically active to try to support your principles. No software can save you from an authoritarian government - you can let that fantasy die.

▲ akerl_ 4 months ago | root | parent | prev | next [–]

I gave you the answer that exists: I'm not aware of any existing or likely-to-exist secure messaging solution that would be a viable recommendation.

The available open-source options come nowhere close to the messaging security that Signal/Whatsapp provide. So you're left with either "find a way to access Signal after they pull out of whatever region has criminalized them operating with a backdoor on comms" or "pick any option that doesn't actually have strong messaging security".

▲ johnisgood 4 months ago | root | parent | next [–]

> messaging security

> WhatsApp

Eh?

There are alternatives, try Ricochet (Refresh) or Cwtch.

▲ akerl_ 4 months ago | root | parent | next [–]

I stand by what I said.

▲ johnisgood 4 months ago | root | parent | next [–]

I mean... why?

▲ closewith 4 months ago | root | parent | next [–]

Not the GP, but most of us want to communicate with other people, which means SMS or WhatsApp. No point have perfect one-time-pad encryption and no one to share pads with.

▲ signed-log 4 months ago | root | parent | prev | next [–]

Most countries will throw you in jail for years if you refuse to give the password to encrypted devices they want. [1]

And that's even if you are innocent on the underlying charge or search.

Encryption in this political climate, is a pick your poison.

- Either you go to jail for years but you know your gov and other actors has no access to your data.

- or you store on remote/proprietary apps, stay free, but your gov or other actors may or may not have access to it.

[1]: https://en.wikipedia.org/wiki/Key_disclosure_law

▲ anonym29 4 months ago | root | parent | prev | next [–]

Could you please link the source code for the WhatsApp client, so that we can see the cryptographic keys aren't being stored and later uploaded to Meta's servers, completely defeating the entire point of Signal's E2EE implementation and ratchet protocol?

▲ akerl_ 4 months ago | root | parent | next [–]

This may shock you, but plenty of cutting-edge application security analysis doesn't start with source code.

There are many reasons, but one of them is that for the overwhelming majority of humans on the planet, their apps aren't being compiled from source on their device. So since you have to account for the fact that the app in the App Store may not be what's in some git repo, you may as well just start with the compiled/distributed app.

▲ anonym29 4 months ago | root | parent | next [–]

Whether or not other people build from source code has zero relevance to a discussion about the trustworthiness of security promises coming from former PRISM data providers about the closed-source software they distribute. Source availability isn't theater, even when most people never read it, let alone build from it. The existence of surreptitious backdoors and dynamic analysis isn't a knock against source availability.

Signal and WhatsApp do not belong in the same sentence together. One's open source software developed and distributed by a nonprofit foundation with a lengthy history of preserving and advancing accessible, trustworthy, verifiable encrypted calling and messaging going back to TextSecure and RedPhone, the other's a piece of proprietary software developed and distributed by a for-profit corporation whose entire business model is bulk harvesting of user data, with a lengthy history of misleading and

manipulating their own users and distributing user data (including message contents) to shady data brokers and intelligence agencies.

To imply these two offer even a semblance of equivalent privacy expectations is misguided, to put it generously.

▲ tptacek 4 months ago | root | parent | next [–]

These are words, but I don't understand how they respond to the preceding comment, which observes that *binary* legibility is an operational requirement for real security given that almost nobody uses reproducible builds. In reality, people meaningfully depend on work done at the binary level to ensure lack of backdoors, not on work done at the source level.

The preceding comment is saying that source security is insufficient, not that transparency is irrelevant.

▲ anonym29 4 months ago | root | parent | next [–]

Source availability is what makes a chain of trust possible that simply isn't meaningfully possible with closed source software, even with dynamic analysis, decompilation, reverse engineering, runtime network analysis with TLS decryption, etc.

Both you and the preceding commenter are correct that just running binaries signed and distributed by Alphabet (Google) and/or Apple presents room for additional risks beyond those observable in the source code, but the solution to this problem isn't to say "and therefore source availability doesn't matter at all for anyone", it's to choose to build from source or to obtain and install APKs built and signed by the developers, such as via Accrescent or Obtanium (pulls directly from github, gitlab, etc releases).

There's a known-good path. Most people do not take the known-good path. Their choice to do so does not invalidate or eliminate the desirable properties of known-good path (verifiability, trustworthiness).

I genuinely do not understand the argument you and the other user are making. It reads to me like an argument that goes "Yes, there's a known, accurate, and publicly documented recipe to produce a cure for cancer, but it requires prerequisite knowledge to understand that most people lack, and it's burdensome to follow the recipe, so most people just buy their vials from the untrustworthy CancerCureCorporation, who has the ability to give customers a modified formula that keeps them sick rather than giving them the actual cure, and almost nobody makes the cure themselves without going through this untrustworthy but ultimately optional intermediary, so the public documentation of the cure doesn't matter at all, and there's no discernable difference between having the cure recipe and not having the cure recipe."

▲ tptacek 4 months ago | root | parent | next [–]

No, you're completely off the rails from the first sentence. It is absolutely possible --- in some ways *more* possible[†] --- to make a chain of trust without source availability. Your premise is that "reverse engineering" is somehow incomplete or lossy with respect to uncovering software behavior, and that simply isn't true.

[†] *Source is always good to have, but it's insufficient.*

▲ anonym29 4 months ago | root | parent | next [–]

Never once anywhere in this thread have I claimed that source code alone is sufficient by itself to establish a chain of trust, merely that it is a necessary prerequisite to establish a chain of trust.

That said, you seem to be refuting even that idea. While your reputation precedes you, and while I haven't been in the field quite as long as you, I do have a few dozen CVEs, I've written surreptitious side channel backdoors and broken production

cryptographic schemes in closed-source software doing binary analysis as part of a red team alongside former NCC folks. I don't know a single one of them who would say that lacking access to source code increases your ability to establish a chain of trust.

Can you please explain how lacking access to source code, being ONLY able to perform dynamic analysis, rather than dynamic analysis AND source code analysis, can ever possibly lead to an increase in the maximum possible confidence in the behavior of a given binary? That sounds like a completely absurd claim to me.

▲ tptacek 4 months ago | root | parent | next [–]

I see what's happening. You're working under the misapprehension that static analysis is only possible with source code. That's not true. In fact: a great deal of real-world vulnerability research is performed statically in a binary setting.

There's a *lot* of background material I'd have to bring in to attempt to bring you up to speed here, but my favorite simple citation here is just: Google [binary lifter].

▲ anonym29 4 months ago | root | parent | next [–]

This assumption about me is not accurate at all, I've done static analysis professionally on CIL, on compiled bytecode, and on source code. Instead of being condescending and patronizing to someone you don't know that you've made factually inaccurate assumptions about, can you please explain how having just a binary and no access to source code gives you more information about, greater confidence in, and a stronger basis for trust in the behavior of a binary than having access to the binary AND the source code used to build it?

▲ tptacek 4 months ago | root | parent | next [–]

I have no idea who you are and can only work from what you write here, and with this comment, what you've written no longer makes sense. The binary (or the lifted IR form of the binary or the control flow graph of the binary or whatever form you're evaluating) is the source of truth about what a program actually does, *not* the source code.

The source code is just a set of hints about what the binary does. You don't need the hints to discern what a binary is doing.

▲ anonym29 4 months ago | root | parent | next [–]

I'm not refuting that the binary is the source of truth about behavior, I never stated it wasn't, and I don't know where you even got the idea that I wasn't. It's been very frustrating to have to repeatedly do this - you and akerl_ have both been attacking strawman positions I do not hold and never stated, and being condescending and patronizing in the process. Is it possible you're making assumptions about me based on arguments made by other people that sound similar to the ones I'm making? I'd really appreciate not having to keep reminding you that I've never made the claims you're implying I'm making, if that's not too much to ask of you.

At a high level, what I'm fundamentally contending is that WhatsApp is less trustworthy and secure

than Signal. I can have a higher degree of confidence in the behavior and trustworthiness of the Signal APK I build from source myself than I can from WhatsApp, which I can't even build a binary of myself. I'd simply be given a copy of it from Google Play or Apple's App Store.

Signal's source code exhibits known trustworthy behavior, i.e. not logging both long-term and ephemeral cryptographic keys and shipping them off to someone else's servers. Sure, Google Play and Apple can modify this source code, add a backdoor, and the binary distributed by Google Play and Apple can have behavior that doesn't match the behavior of the published source code. You can detect this fairly easily, because you have a point of reference to compare to. You know what the compiled bytecode from the source code you've reviewed looks like, because you can build it yourself, no

trust required[1], it's not difficult to see when that differs in another build.

With WhatsApp, you don't even have a point of reference of known good behavior, i.e. not logging both long-term and ephemeral cryptographic keys and shipping them off to someone else's server, in the first place. You can monitor all the disk writes, you can monitor all the network activity. Just because YOU don't observe cryptographic keys being logged, either in-memory, or on disk, or being sent off to some other server, doesn't mean there isn't code present to perform those exact functions under conditions you've never met and never would - it's entirely technically feasible for Google and Apple to be fingerprinting a laundry list of identifiers of known security researchers and be shipping them binaries with behavior that differs from the behavior of ordinary users, or even for them to ship targeted backdoored binaries to

specific users at the demand of various intelligence agencies.

The upper limit for the trustworthiness of a Signal APK you build from source yourself is on a completely different planet from the trustworthiness of a WhatsApp APK you only have the option of receiving from Google.

And again, none of this even begins to factor in Meta's extensive track record on deliberately misleading users on privacy and security through deceptive marketing and subverting users' privacy extensively. Onavo wasn't just capturing all traffic, it was literally doing MITM attacks against other companies' analytics servers with forged TLS certificates. Meta was criminally investigated for this and during discovery, it came out that executives understood what was going on, understood how wrong it was, and deliberately continued with the practice anyway.

Actual technical analysis of the binaries and source code aside, it's plainly ridiculous to suggest that software made by that same corporation is as trustworthy as Signal. One of these apps is a messenger made by a company with a history of explicitly misleading users with deceptive privacy claims and employing non-trivial technical attacks against their own users to violate their own users' privacy, the other is made by a nonprofit with a track record of being arguably one of the single largest contributors to robust, accessible, audited, verifiable secure cryptography in the history of the field. I contend that suggesting these two applications are equally secure is irrational, impossible to demonstrate or verify, and indefensible.

[1] Except in your compiler, linker, etc... Ken Thompson's 'Reflections on Trusting Trust' still applies here. The argument isn't that source code availability automatically

means 100% trustworthy, it means the upper boundary for trustworthiness is higher than without source availability.

▲ akerl_ 4 months ago | root | parent | next [–]

It's clear we're not going to agree on the technical discussion, but I do want to reply to the claim that I've been strawmanning you.

I've been largely ignoring your sideline commentary about not trusting Meta and their other work outside of WhatsApp. Mostly because the whole thrust of my argument is that an

app's security is confirmed by analyzing what the code does, not by listening to claims from the author.

Beyond that, I've been commenting in good faith about the core thrust of our disagreement, which is whether or not a lack of available source code disqualifies WhatsApp as a viable secure messaging option alongside Signal.

As part of that, I had to respond midway through because you put a

statement in quotation marks that was not actually something I'd said.

▲ tptacek 4 months ago | root | parent | prev | next [–]

Sorry, no, I'm not going to pick this apart. You wrote:

Can you please explain how lacking access to source code, being ONLY able to perform dynamic analysis, rather than dynamic analysis AND source code analysis, can ever possibly lead to an increase in the maximum

possible confidence in the behavior of a given binary?

This doesn't make sense, because not having source code doesn't limit you to dynamic analysis. I assumed, 2 comments back, you were just misunderstanding SOTA reversing; you got mad at me about that. But the thing you "never stated it wasn't" is right there in the comment history. Acknowledge that and help me understand where the

gap was, or this isn't worth all the words you're spending on it.

▲ akerl_ 4 months ago | root | parent | prev | next [–]

> but the solution to this problem isn't to say "and therefore source availability doesn't matter at all for anyone"

Thankfully, I didn't say that.

▲ anonym29 4 months ago | root | parent | next [–]

Great, then it sounds like we agree: your original equivalence of Signal and WhatsApp was misguided, since one offers a verifiable chain of trust that starts with source availability and the other doesn't, to say nothing of the lengthy history of untrustworthiness and extensive, deliberate privacy violations of the company that owns and maintains WhatsApp, right?

▲ akerl_ 4 months ago | root | parent | next [–]

No, we don't agree. There are things that source code is good for, but validating the presence or absence of illicit data stealing code in apps delivered to consumers is not one of those things. For that, source code can show you obvious malfeasance, but since it's not enough to rule out obvious malfeasance, you're stuck going to analysis of the compiled app in both cases.

The population of users who have a verifiable path from an open source repo to an app on their device is a rounding error in the set of humans using messaging apps.

▲ anonym29 4 months ago | root | parent | next [–]

I think we've both made our positions clear. From my perspective, you're continuing to heavily cite user statistics that are irrelevant to the properties of verifiability or trustworthiness of the applications themselves, the goalposts I am discussing keep being moved, and there is a repeated pattern of neglect to address the points I'm raising. Readers can judge for themselves. Curious readers should also read about the history of Meta's Onavo VPN software and resulting lawsuits and settlements in evaluating the credibility of Meta's privacy marketing.

▲ akerl_ 4 months ago | root | parent | next [–]

Just to be crystal clear about the goalposts: I said at the start of this chain that if somebody wants secure messaging, they should use Signal or WhatsApp.

You raised concerns about lack of source availability, and I've been consistent in my replies that source availability is not the way that somebody wants secure messaging is going to know they're getting it. They're going to get it because they're using a popular platform with robust primitives, whose compiled/distributed apps receive constant scrutiny from security researchers.

Signal and WhatsApp are that. Concerns about Meta's other work are just noise, in part because analysis of the WhatsApp distributed binaries doesn't rely on promises from Meta.

▲ johnisgood 4 months ago | root | parent | prev | next [–]

Saw it, not impressed, GnuPG has a lot of more features than signing and file encryption.

And there are lots of tools for file encryption anyways. I have a bash function using openssh, sometimes I use croc (also uses PAKE), etc.

I need an alternative to "gpg --encrypt --armor --recipient <foo>". :)

▲ akerl_ 4 months ago | root | parent | next [–]

I guess we'll have to live with you being unimpressed.

▲ some_furry 4 months ago | root | parent | prev | next [–]

> I need an alternative to "gpg --encrypt --armor --recipient <foo>"

That's literally age.

https://github.com/FiloSottile/age

▲ johnisgood 4 months ago | root | parent | next [–]

No, because there is no keyring and you have to supply people's public key each time. It is not suitable for large-scale public key management (with unknown recipients), and it does not support automatic discovery, trust management. Age does NOT SUPPORT signing at all either.

▲ some_furry 4 months ago | root | parent | next [–]

Why is a keyring important to you?

Would "fetch a short-lived age public key" serve your use case? If so, then an age plugin that build atop the AuxData feature in my Fediverse Public Key Directory spec might be a solution. https://github.com/fedi-e2ee/public-key-directory-specificat...

But either way, you shouldn't have long-lived public keys used for confidentiality. It's a bad design to do that.

▲ johnisgood 4 months ago | root | parent | next [–]

> you shouldn't have long-lived public keys used for confidentiality.

This statement is generic and misleading. Using long-lived keys for confidentiality is bad in real-time messaging, but for non-ephemeral use cases (file encryption, backups, archives) it is completely fine AND desired.

> Would "fetch a short-lived age public key" serve your use case?

Sadly no.

▲ soatok 4 months ago | root | parent | next [–]

(This is some_furry, I'm currently rate-limited. I thought this warranted a reply, so I switched to this account to break past the limit for a single comment.)

> This statement is generic and misleading.

It may be generic, but it's not misleading.

> Using long-lived keys for confidentiality is bad in real-time messaging, but for non-ephemeral use cases (file encryption, backups, archives) it is completely fine.

What exactly do you mean by "long-lived"?

The "lifetime" of a key being years (for a long-lived backup) is less important than how many encryptions are performed with said key.

The thing you don't want is to encrypt 2^50 messages under the same key. Even if it's cryptographically safe to do that, any post-compromise key rotation will be a fucking nightmare.

The primary reason to use short-lived public keys is to limit the blast radius. Consider these two companies:

Alice Corp. uses the same public key for 30+ years.

Bob Ltd. uses a new public key for each quarter over the same time period.

Both parties might retain the secret key indefinitely, so that if Bob Ltd. needs to retrieve a backup from 22 years ago, they still can.

Now consider what happens if both of them lose their currently-in-use secret key due to a Heartbleed-style attack. Alice has 30 years of disaster recovery to contend with, while Bob only has up to 90 days.

Additionally, file encryption, backups, and archives typically use *ephemeral symmetric keys* at the bottom of the protocol. Even when a password-based key derivation function is used (and passwords are, for whatever reason, reused), the password hashing function usually has a random salt, thereby guaranteeing uniqueness.

The idea that "backups" magically mean "long-lived" keys are on the table, without nuance, is extremely misleading.

> > Would "fetch a short-lived age public key" serve your use case?

> Sadly no.

shrug Then, ultimately, there is no way to securely satisfy your use case.

▲ johnisgood 4 months ago | root | parent | next [–]

You introduced "short-lived" vs "long-lived", not me. Long-lived as wall-clock time (months, years) is the default interpretation in this context.

The Alice / Bob comparison is asymmetric in a misleading way. You state Bob Ltd retains all private keys indefinitely. A Heartbleed-style attack on their key storage infrastructure still compromises 30 years of backups, not 90 days. Rotation only helps if *only* the current operational key is exposed, which is an optimistic threat model you did not specify.

Additionally, your symmetric key point actually supports what I said. If data is encrypted with ephemeral symmetric keys and the asymmetric key only wraps those, the long-lived asymmetric key's exposure does not enable bulk decryption without obtaining each wrapped key individually.

> "There is no way to securely satisfy your use case"

No need to be so dismissive. Personal backup encryption with a long-lived key, passphrase-protected private key, and offline storage is a legitimate threat model. Real-world systems validate this: SSH host keys, KMS master keys, and yes, even PGP, all use long-lived asymmetric keys for confidentiality in non-ephemeral contexts.

And to add to this, incidentally, age (the tool you mentioned) was designed with long-lived recipient keys as the expected use case. There is no built-in key rotation or expiry mechanism because the authors considered it unnecessary for file encryption. If long-lived keys for confidentiality were inherently problematic, age would be a flawed design (so you might want to take it up with them, too).

In any case, yeah, your point about high-fan-out keys with large blast radius is correct. That is different from "long-lived keys are bad for confidentiality" (see above with regarding to "age").

▲ maxtaco 4 months ago | root | parent | next [–]

An intended use case for FOKS (https://foks.pub) is to allow long-lived durable shared secrets between users and teams with key rotation when needed.

▲ some_furry 4 months ago | root | parent | prev | next [–]

> The Alice / Bob comparison is asymmetric in a misleading way. You state Bob Ltd retains all private keys indefinitely. A Heartbleed-style attack on their key storage infrastructure still compromises 30 years of backups, not 90 days.

No. Having 30 years of secret keys *at all* is not the same of having 30 years of secret keys *in memory*.

▲ stackghost 4 months ago | root | parent | prev | next [–]

>Personal backup encryption with a long-lived key, passphrase-protected private key, and offline storage is a legitimate threat model

... If you're going to use a passphrase anyway why not just use a symmetric cipher?

In fact for file storage why not use an encrypted disk volume so you don't need to use PGP?

▲ johnisgood 4 months ago | root | parent | next [–]

That was just me being goofy in that bit (and only that), but I hope the rest of my message went across. :)

> In fact for file storage why not use an encrypted disk volume so you don't need to use PGP?

Different threat models. Disk encryption (LUKS, VeraCrypt, plain dm-crypt) protects against physical theft. Once mounted, everything is plaintext to any process with access. File-level encryption protects files at rest and in transit: backups to untrusted storage, sharing with specific recipients, storing on systems you do not fully control. You cannot send someone a LUKS volume to decrypt one file, and backups of a mounted encrypted volume are plaintext unless you add another layer.

▲ stackghost 4 months ago | root | parent | next [–]

>You cannot send someone a LUKS volume to decrypt one file, and backups of a mounted encrypted volume are plaintext unless you add another layer.

Veracrypt, and I'm sure others, allow you to do exactly this. You can create a disk image that lives in a file (like a .iso or .img) and mount/unmount it, share it, etc.

▲ akerl_ 4 months ago | root | parent | next [–]

That's not what they said. They're saying you often want to give someone a specific file from a disk, rather than the whole set of files.

▲ stackghost 4 months ago | root | parent | next [–]

You can still do that with a .dmg, for example. I've done it, it works more or less like a zip.

But even if that was somehow unreasonable or undesired, you can use Filippo's age for that. PGP has no use case that isn't done better by some other tool, with the possible exception of "cosplay as a leet haxor"

▲ deknos 4 months ago | root | parent | prev | next [–]

We need a keyring at a company. Because there's no other media for communicating, where you reach management and technical people in companies as well.

And we have massive issues due to the fact that the ongoing-decrying of "shut everything off" and the following non-improvement-without-an-alternative because we have to talk with people of other organizations (and every organization runs their own mailserver) and the only really common way of communication is Mail.

And when everyone has a GPG Key, you get.. what? an keyring.

You could say, we do not need gpg, because we control the mailserver, but what if a mailserver is compromised and the mails are still in mailboxes?

the public keys are not that public, only known to the contenders, still, it's an issue and we have a keyring

▲ Natanael_L 4 months ago | root | parent | next [–]

You need a private PKI, not keyring. They're subtly different - a PKI can handle key rotation, etc.

Yes there aren't a lot of good options for that. If you're using something like a Microsoft software stack with active directory or similar identity/account management then there's usually some PKI support in there to anchor to.

Across organisations, there's really very very few good solutions. GPG specifically is much too insecure when you need to receive messages from untrusted senders. There's basically S/MIME which have comparable security issues, then we have AD federation or Matrix.org with a server per org.

> You could say, we do not need gpg, because we control the mailserver, but what if a mailserver is compromised and the mails are still in mailboxes?

How are you handling the keys? This is only true if user's protect their own keypairs with strong passwords / yubikey applet, etc.

▲ some_furry 4 months ago | root | parent | prev | next [–]

> We need a keyring at a company.

https://xyproblem.info

Look closely at the UX I'm proposing in https://github.com/fedi-e2ee/pkd-client-php?tab=readme-ov-fi...

Tell me why this won't work for your company.

▲ amluto 4 months ago | root | parent | prev | next [–]

> you have to supply people's public key each time

Keyrings are awful. I *want* to supply people's public keys each time. I have never, in my entire time using cryptography, wanted my tool to guess or infer what key to verify with. (Heck, JOSE has a long history of bugs because it infers the key *type*, which is also a mistake.)

I have an actual commercial use case that receives messages (which are, awkwardly, files sent over various FTP-like protocols, sigh), decrypts and verifies them, and further processes them. This is fully automated and runs as a service. For horrible legacy reasons, the files are in PGP format. I know the public key with which they are signed (provisioned out of band) and I have the private key for decryption (again, provisioned out of band).

This would be approximately two lines of code using any sane crypto library [0], but there really isn't an amazing GnuPG alternative that's compatible enough.

But GnuPG has keyrings, and it really wants to use them and to find them in some home directory. And it wants to identify keys by 32-bit truncated hashes. And it wants to use Web of Trust. And it wants to support a zillion awful formats from the nineties using wildly insecure C code. All of this is actively counterproductive. Even ignoring potential implementation bugs, I have far more code to deal with key *rings* than actual gpg invocation for useful crypto.

[0] I should really not have to even think about the interaction between decryption and verification. Authenticated decryption should be one operation, or possibly two. But if it's two, it's one operation to decapsulate a session key and a second operation to perform authenticated decryption using that key.

▲ mkesper 4 months ago | root | parent | next [–]

Some years ago I wrote "just a little script" to handle encrypting password-store secrets for multiple recipients. It got quite ugly and much more verbose than planned, switching gpg output parsing to Python for sanity. I think I used a combination of --keyring <mykeyring> --no-default-keyring. Never would encourage anyone to do this again.

▲ upofadown 4 months ago | root | parent | prev | next [–]

>And it wants to identify keys by 32-bit truncated hashes.

That's 64 bits these days.

>I should really not have to even think about the interaction between decryption and verification.

Messaging involves two verifications. One to insure that you are sending the message to who you think you are sending the message. The other to insure that you know who you received a message from. That is an inherent problem. Yes, you can use a shared key for this but then you end up doing both verifications manually.

▲ amluto 4 months ago | root | parent | next [–]

>> And it wants to identify keys by 32-bit truncated hashes.

> That's 64 bits these days.

The fact that it's short enough that I even need to think about whether it's a problem is, frankly, pathetic.

> Messaging involves two verifications. One to insure that you are sending the message to who you think you are sending the message. The other to insure that you know who you received a message from. That is an inherent problem. Yes, you can use a shared key for this but then you end up doing both verifications manually.

I can't quite tell what you mean.

One can build protocols that do encrypt-then-sign, encrypt-and-sign, sign-then-encrypt, or something clever that combines encryption and signing. Encrypt-then-sign has a nice security proof, the other two combinations are often somewhat catastrophically wrong, and using a high quality combination can have good performance and nice security proofs.

But all of the above should be the job of the designer of a protocol, not the user of the software. If my peer sends me a message, I should provision keys, and then I should pass those keys to my crypto library along with a message I received (and perhaps whatever session state is needed to detect replays), and my library should either (a) tell me that the message is invalid and *not* give me a guess as to its contents or (b) tell me it's valid and give me the contents. I should not need to separately handle decryption and verification, and I should not even be able to do them separately even if I want to.

▲ upofadown 4 months ago | root | parent | next [–]

>The fact that it's short enough that I even need to think about whether it's a problem is, frankly, pathetic.

Please resist the temptation to personally attack others.

I think you mean that 64 bits of hash output could be trivially collided using, say, Pollard's rho method. But it turns out that simple collisions are not an issue for such hashes used as identities. The fact that PGP successfully used 32 bits (16 bits of effort for a collision) for so long is actually a great example of the principle.

>...encrypt-then-sign, encrypt-and-sign, sign-then-encrypt...

You mean encrypt-then-MAC here I think.

>...I should not even be able to do them separately even if I want to.

Alas that is not possible. The problem is intrinsic to end to end encrypted messaging. Protocols like PGP combine them into a single key fingerprint so that the user does not have to deal with them separately. You still have to verify the fingerprint for people you are sending to and the fingerprint for the people who send you messages.

▲ tptacek 4 months ago | root | parent | next [–]

They didn't personally attack you. They (correctly) attacked 64-bit identifiers.

▲ upofadown 4 months ago | root | parent | next [–]

They were attacking an entire community. Perhaps I should have complained about being deliberately provocative.

But to the point, how long should something like a key fingerprint be?

▲ some_furry 4 months ago | root | parent | next [–]

> How long should something like a key fingerprint be?

At least 128 bits for most threat models. 192+ is preferable for mine.

https://soatok.blog/2024/07/01/blowing-out-the-candles-on-th...

My threat model assumes you want an attacker advantage of less than 2^-64 after 2^64 keys exist to be fingerprinted in the first place, and your threat model includes collisions.

If I remember correctly, cloud providers assess multi-user security by assuming 2^40 users which each will have 2^50 keys throughout their service lifetime.

If you round down your assumption to 2^34 users with at most 100 public keys on average (for a total of 2^41 user-keys), you can get away with 2^-41 after 2^41 at about 123 bits, which for simplicity you can

round up to the nearest byte and arrive at 128 bits.

The other thing you want to keep in mind is, how large are the keys in scope? If you have 4096-bit RSA keys and your fingerprints are only 64 bits, then by the pigeonhole principle we expect there to be 2^4032 distinct public keys with a given fingerprint. The average distance between fingerprints will be random (but you can approximate it to be an order of magnitude near 2^32).

In all honesty, fingerprints are probably a poor mechanism.

▲ upofadown 4 months ago | root | parent | next [–]

>...and your threat model includes collisions.

OK, to be clear, I am specifically contending that a key fingerprint does *not* include collisions. My proof is empirical, that no one has come up with an attack on 64 bit PGP key fingerprints.

Collisions mean that an attacker can generate two or more messaging identities with the same fingerprint. How would that help them in some way?

▲ some_furry 3 months ago | root | parent | next [–]

https://soatok.blog/2026/01/07/practical-collision-attack-ag...

Q.E.D.

▲ upofadown 3 months ago | root | parent | next [–]

Sorry that my, perhaps, poor wording caused you to waste your time producing colliding 64 bit PGP key IDs. I should have used the term "threat model". We were discussing how long key fingerprints should be. My point was that even though 64 bit key IDs are trivially collidable there did not seem to be any practical attacks based

on that. So you in a sense provided support for my argument. :) So we can skip directly to your proposed attack...

I have to admit that I don't actually understand it. First the attacker gets some kernel devs to sign key1 of the two keys with colliding key IDs. Why? How does that help the attacker? Then I am guessing that the attacker signs some software with key1. Are the signatures important here? Then the attacker signs the malicious software with key2? Key2 isn't signed by any developers so if that was important the attack fails. If it wasn't important then why mention it?

Could you please provide a more detailed description of the attack? It seems to me that the sort of attack you are describing would require some trusted third party to trick. Like a TLS certifying authority for example.

▲ amluto 4 months ago | root | parent | prev | next [–]

> attacker advantage of less than 2^-64

Why so high? Computers are fast and massively parallel these days. If a cryptosystem fully relies on

fingerprints, a second preimage of someone's fingerprint where the attacker knows the private key for the second preimage (or it's a cleverly corrupt key pair) catastrophically breaks security for the victim. Let's make this astronomically unlikely even in the multiple potential victim case.

And it's not like 256 bit hashes are expensive.

(I'm not holding my breath on fully quantum attacks using Grover's algorithm, at high throughput, against billions of users, so we can probably wait a while before 256 bits feels uncomfortably short.)

▲ upofadown 4 months ago | root | parent | next [–]

>And it's not like 256 bit hashes are expensive.

A key fingerprint is a usability feature. It has no other purpose. Otherwise we would just use the public key. Key fingerprints have to be kept as short as possible. So the question is, how short can that be? I would argue that 256 bit key fingerprints are not really usable.

Signal messenger is using 100 bits for their key fingerprint. They combine two to make a 60 digit decimal number. Increasing that to 256 x 2 bits would mean that they would end up with 154 decimal digits. That would be completely unusable.

▲ some_furry 4 months ago | root | parent | prev | next [–]

I was asked about the minimum value, and gave my explanation for why some values could be considered the minimum. By all means, use 256-bit fingerprints.

▲ tptacek 4 months ago | root | parent | prev | next [–]

No, again, they were attacking 64-bit identifiers.

▲ amluto 4 months ago | root | parent | prev | next [–]

> I think you mean that 64 bits of hash output could be trivially collided using, say, Pollard's rho method. But it turns out that simple collisions are not an issue for such hashes used as identities.

No. I mean that 64 bits can probably be inexpensively attacked to produce first or second preimages.

It would be nice if a decentralized crypto system had memorable key identifiers and remained secure, but I think that is likely to be a pipe dream. So a tool like gpg shouldn't even try. Use at least 128 bits and give three choices: identify keys by an actual secure hash or identify them by a name the user assigns or pass them directly. Frankly I'm not sure why identifiers are even useful — see my original complaint about keyrings.

>> ...I should not even be able to do them separately even if I want to.

>Alas that is not possible. The problem is intrinsic to end to end encrypted messaging. Protocols like PGP combine them into a single key fingerprint so that the user does not have to deal with them separately.

Huh? It's possible. It's not even hard. It could work like this:

$ better_gpg decrypt_and_auth --sender_pubkey [KEY] --recipient_privkey [KEY]

Ciphertext input is supplied on stdin. Plaintext output appears on stdout but only if the message validates correctly.

▲ upofadown 4 months ago | root | parent | next [–]

>I mean that 64 bits can probably be inexpensively attacked to produce first or second preimages.

Keep in mind that you would have to generate a valid keypair, or something that could be made into a valid keypair for each iteration. That fact is why PGP got along with 32 bit key IDs for so long. PGP would still be using 32 bit key IDs if it wasn't that someone figured out how to mess with RSA exponents to greatly speed up the process. Ironically, the method with the slowest keypair generation became the limiting factor.

It isn't like this is a new problem. People have been designing and using key fingerprint schemes for over a quarter of a century now.

>$ better_gpg decrypt_and_auth --sender_pubkey [KEY] --recipient_privkey [KEY]

How do you know that the recipient key actually belongs to the recipient? How does the recipient know that the sender key actually belongs to you (so it will validate correctly)?

▲ Natanael_L 4 months ago | root | parent | prev | next [–]

What you described IS WHY age is the better option.

GPG's keyring handling has also been a source of exploits. It's much safer to directly specify recipient rather than rely on things like short key IDs which can be bruteforced.

Automatic discovery simply isn't secure if you don't have an associated trust anchor. You need something similar to keybase or another form of PKI to do that. GPG's key servers are dangerous.

You technically can sign with age, but otherwise there's minisign and the SSH spec signing function

▲ pseudohadamard 4 months ago | root | parent | prev | next [–]

And when do you need any of that stuff?

As a followup, is there anything in existence that supports "large-scale public key management (with unknown recipients)"? Or "automatic discovery, trust management"? Even X.509 PKI at its most delusional doesn't claim to be able to do that.

▲ baobun 4 months ago | root | parent | prev | next [–]

sq (sequoia) should be able to sort that.

▲ johnisgood 4 months ago | root | parent | next [–]

I know, I have been using it recently.

▲ benchloftbrunch 4 months ago | root | parent | prev | next [–]

What is the alternative to PGP for the specific use case of secure email? That doesn't mandate dealing with the X509 certificate bureaucracy?

▲ tptacek 4 months ago | root | parent | next [–]

Don't encrypt email.

https://www.latacora.com/blog/2020/02/19/stop-using-encrypte...

▲ teddyh 4 months ago | root | parent | next [–]

The only alternative suggested by the linked article is giving up email completely in favor of centralized solutions like Signal. My short answer is “no”. My long answer is: <https://news.ycombinator.com/item?id=45390332>

▲ tptacek 4 months ago | root | parent | next [–]

I wrote the linked article. I don't care what secure messenger you use. But if you choose encrypted email over Signal because "centralization", you're LARPing. The first criteria for a secure messenger has to be that it is plausibly secure, and email isn't. You'd use encrypted email (for "decentralization") because you understand the cost of losing the plaintext of your message is nil. If you tell strangers to do that, without certainty that their messages are also valueless, you're committing malpractice.

▲ pseudohadamard 4 months ago | root | parent | prev | next [–]

Something that doesn't require securing email. Both S/MIME and PGP were solutions for 1980s problems (TFA is slightly off about PGP's start date, the PGP design dates from 1987 and MSDOS, not the 1990s, and S/MIME via PEM is from 1986). They're pretty much irrelevant today because almost all email is encrypted anyway via StartTLS and if you need full end-to-end encryption you use Signal or something similar.

▲ Natanael_L 4 months ago | root | parent | prev | next [–]

What's your usecase here? Internal or external messaging?

▲ pseudohadamard 4 months ago | root | parent | next [–]

Use case? We're crypto LARPing dammit, we don't need a use case!

▲ josephg 4 months ago | root | parent | prev | next [–]

The thing I can't get past with PGP / GPG is that it tries to work around MITM attacks by encouraging users to place their social network on the public record (via public key attestation).

This is so insane to me. The whole point of using cryptography is to keep private information private. Its hard to think of ways PGP could fail more as a security / privacy tool.

▲ upofadown 4 months ago | root | parent | next [–]

Do you mean keyservers? Keyservers have nothing to do with the identity verification required to prevent MITM attacks. There is only one method available for PGP. Comparison of key fingerprints/IDs.

Keyservers are simply a convenient way to get a public key (identity). Most people don't have to use them.

▲ coppsilgold 4 months ago | root | parent | prev | next [–]

Depending on what you are after, an alternative could be using SSH keys for signatures and age[1] for encryption targeting SSH keys.

[1] <https://github.com/FiloSottile/age>

▲ baobun 4 months ago | root | parent | prev | next [–]

sq (sequoia) is compatible and is available in your favorite distro. It's the recommended replacement.

https://book.sequoia-pgp.org/about_sequoia.html

▲ zimmerfrei 4 months ago | root | parent | next [–]

This is the right answer.

The problem mostly concerns the oldest parts of PGP (the protocol), which gpg (the implementation) doesn't want or cannot get rid of.

▲ lagniappe 4 months ago | root | parent | prev | next [–]

age https://github.com/FiloSottile/age

▲ vbezhenar 4 months ago | root | parent | prev | next [–]

age

▲ alphazard 4 months ago | parent | prev | next [–]

It's a fundamentally bad idea to have a single key that applications are supposed to look for in a particular place, and then use to sign things. There is inherent complexity involved in making multi-context key use safe, and it's better to just avoid it architecturally.

Keys (even quantum safe) are small enough that having one per application is not a problem at all. If an application needs multi-context, they can handle it themselves. If they do it badly, the damage is contained to that application. If someone really wants to make an application that just signs keys for other applications to say "this is John Smith's key for git" and "this is John Smith's key for email" then they could do that. Such an application would not need to concern itself with permissions for other applications calling into it. The user could just copy and paste public keys, or fingerprints when they want to attest to their identity in a specific application.

The keyring circus (which is how GPG most commonly intrudes into my life) is crazy too. All these applications insist on connecting to some kind of GPG keyring instead of just writing the secrets to the filesystem in their own local storage. The disk is fully encrypted, and applications should be isolated from one another. Nothing is really being accomplished by requiring the complexity of yet another program to "extra encrypt" things before writing them to disk.

I'm sure these bad ideas come from the busy work invented in corporate "security" circles, which invent complexity to keep people employed without any regard for an actual threat model.

▲ akerl_ 4 months ago | root | parent | next [–]

> The disk is fully encrypted, and applications should be isolated from one another.

For most apps on non-mobile devices, there isn't filesystem isolation between apps. Disk/device-level encryption solves for a totally different threat model; Apple/Microsoft/Google all ship encrypted storage for secrets (Keychain, Credential Manager, etc), because restricting key material access within the OS has merit.

> I'm sure these bad ideas come from the busy work invented in corporate "security" circles, which invent complexity to keep people employed without any regard for an actual threat model.

Basically everything in PGP/GPG predates the existence of "corporate security circles".

▲ Avamander 4 months ago | root | parent | next [–]

> For most apps on non-mobile devices, there isn't filesystem isolation between apps.

If there isn't there should be. At least my Flatpaks are isolated from each other.

> Apple/Microsoft/Google all ship encrypted storage for secrets (Keychain, Credential Manager, etc), because restricting key material access within the OS has merit.

The Linux equivalents are suspicious and stuck in the past to say the least. Depending on them is extra tedious on top of the tediousness of any PGP keyrings, god forbid a combination of the two.

> Basically everything in PGP/GPG predates the existence of "corporate security circles".

Then we know where this stuff came from.

▲ akerl_ 4 months ago | root | parent | next [–]

> Then we know where this stuff came from.

I can’t figure out what you mean by this.

▲ Avamander 4 months ago | root | parent | next [–]

Just a joke that if indeed GPG predates and was not inspired by corporate security theatre then the opposite must be true. That corporate security theatre was inspired by GPG/PGP.

▲ deknos 4 months ago | root | parent | prev | next [–]

and now certain people in corporate security only trust gpg, because they grew up with it :D

▲ xorcist 4 months ago | parent | prev | next [–]

These are not vulnerabilities in the "remote exploit" sense. They should be taken seriously, you should be careful not to run local software on untrusted data, and GPG should probably do more to protect users from shooting themselves in the foot, but the worst thing you could do is panic and throw out a process your partners and colleagues trust. There is nothing here that will disturb your workflow signing commits or apt-get install-ing from your distribution.

If you use crypographic command line tools to verify data sent to you, be mindful on what you are doing and make sure to understand the attacks presented here. One of the slides is titled "should we even use command line tools" and yes, we should because the alternative is worse, but we must be diligent in treating all untrusted data as adversarial.

▲ akerl_ 4 months ago | root | parent | next [–]

A huge part of GPG's purported use case is getting a signed/encrypted/both blob from somebody and using GPG to confirm it's authentic. This is true for packages you download and for commits with signatures.

Handling untrusted input is core to that.

▲ xorcist 4 months ago | root | parent | next [–]

It is, and other software handling untrusted data should also treat it as adversarial. For example, your package tool should probably not output raw package metadata to the terminal.

▲ akerl_ 4 months ago | root | parent | next [–]

I think you're missing the forest for the trees.

▲ tgsovlerkhgsel 4 months ago | root | parent | prev | next [–]

It reads to me like attempting to verify a malicious ascii-armoured signature is potential RCE.

▲ larusso 4 months ago | parent | prev | next [–]

I did the switch this year after getting yet another personal computer. I have 4 in total (work laptop, personal sofa laptop, Mac Mini, Linux Tower). I used Yubi keys with gpg and resident ssh keys. All is fine but the configuration needed to get it too work on all the machines. I also tend to forget the finer details and have to relearn the skills of fetching the public keys into the keychain etc. I got rid of this all by moving to 1Password ssh agent and git ssh signing. Removes a lot of headaches from my ssh setup. I still have the yubi key(s) though as a 2nd factor for certain web services. And the gpg agent is still running but only as a fallback. I will turn this off next year.

▲ snorremd 4 months ago | root | parent | next [–]

I've ended up the same place as you. I had previously set up my gpg key on a Yubikey and even used that gpg key to handle ssh authentication. Then at some point it just stopped working, maybe the hardware on my key broke. 2FA still works though.

In any case I figured storing an SSH key in 1Password and using the integrated SSH socket server with my ssh client and git was pretty nice and secure enough. The fact the private key never leaves the 1Password vault unencrypted and is synced between my devices is pretty neat. From a security standpoint it is indeed a step down from having my key on a physical key device, but the hassle of setting up a new Yubikey was not quite worth it.

I'm sure 1Password is not much better than having a passphrase-protected key on disk. But it's a lot more convenient.

▲ DetectDefect 4 months ago | root | parent | next [–]

> I had previously set up my gpg key on a Yubikey and even used that gpg key to handle ssh authentication. Then at some point it just stopped working, maybe the hardware on my key broke

Did you try to SSH in verbose mode to ascertain any errors? Why did you assume the hardware "broke" without anyone objective qualifications of an actual failure condition?

> I figured storing an SSH key in 1Password and using the integrated SSH socket server with my ssh client and git was pretty nice and secure enough

How is trusting a closed-source, for-profit, subscription-based application with your SSH credential "secure enough"?

Choosing convenience over security is certainly not unreasonable, but claiming both are achieved without any compromise borders on ludicrous.

▲ hirako2000 4 months ago | root | parent | prev | next [–]

How is 1password safer than the local keychain?

▲ larusso 4 months ago | root | parent | next [–]

The keys never leave the 1Password store. So you don't have the keys on the local file system. That and that these keys are shared over the cloud was the seller for me. I guess security wise it's a bit of a downgrade compared to resident keys. But the agent support agent forwarding etc which wasn't really working with yubi ssh resident keys. Also worth mentioning that I use 1Password. Bitwarden has a similar feature as far as I know. For the ones who want to self host etc might be the even better solution.

▲ akerl_ 4 months ago | root | parent | next [–]

> The keys never leave the 1Password store. So you don't have the keys on the local file system.

Keychain and 1Password are doing variants of the same thing here: both store an encrypted vault and then give you credentials by decrypting the contents of that vault.

▲ hk1337 4 months ago | root | parent | prev | next [–]

> 1Password ssh agent and git ssh signing

I'm still working through how to use this but I have it basically setup and it's great!

▲ 65a 4 months ago | parent | prev | next [–]

> I certainly want to get rid of gpg from my life if I can

I see this sentiment a lot, but you later hint at the problem. Any "replacement" needs to solve for secure key distribution. Signing isn't hard, you can use a lot of different things other than gpg to sign something with a key securely. If that part of gpg is broken, it's a bug, it can/should be fixed.

The real challenge is distributing the key so someone else can verify the signature, and almost every way to do that is fundamentally flawed, introduces a risk of operational errors or is annoying (web of trust, trust on first use, central authority, in-person, etc). I'm not convinced the right answer here is "invent a new one and the ecosystem around it".

▲ akerl_ 4 months ago | root | parent | next [–]

It's not like GPG solves for secure key distribution. GPG keyservers are a mess, and you can't trust their contents anyways unless you have an out of band way to validate the public key. Basically nobody is using web-of-trust for this in the way that GPG envisioned.

This is why basically every modern usage of GPG either doesn't rely on key distribution (because you already know what key you want to trust via a pre-established channel) or devolves to the other party serving up their pubkey over HTTPS on their website.

▲ 65a 4 months ago | root | parent | next [–]

Yes, not saying that web of trust ever worked. "Pre-established channel" are the other mechanisms I mentioned, like a central authority (https) or TOFU (just trust the first key you get). All of these have some issues, that any alternative must also solve for.

▲ akerl_ 4 months ago | root | parent | next [–]

So if we need a pre-established channel anyways, why would people recommending a replacement for GPG workflows need to solve for secure key distribution?

This is a bit like looking at electric cars and saying ~"well you can't claim to be a viable replacement for gas cars until you can solve flight"

▲ woodruffw 4 months ago | root | parent | prev | next [–]

A lot of people are using PGP for things that don't require any kind of key distribution. If you're just using it to encrypt files (even between pointwise parties), you can probably just switch to age.

(We're also *long* past the point where key distribution has been a significant component of the PGP ecosystem. The PGP web of trust and original key servers have been dead and buried for years.)

▲ kaoD 4 months ago | root | parent | prev | next [–]

This is not the first time I see "secure key distribution" mentioned in HN+(GPG alternatives) context and I'm a bit puzzled.

What do you mean? Web of Trust? Keyservers? A combination of both? Under what use case?

▲ kpil 4 months ago | root | parent | next [–]

I'm assuming they mean the old way of signing each others signatures.

As a practical implementation of "six degrees of Kevin Bacon", you could get an organic trust chain to random people.

Or at least, more realistically, to few nerds. I think I signed 3-4 peoples signatures.

The process had - as they say - a low WAF.

▲ dale_glass 4 months ago | root | parent | next [–]

> As a practical implementation of "six degrees of Kevin Bacon", you could get an organic trust chain to random people.

GPG is terrible at that.

0. Alice's GPG trusts Alice's key tautologically. 1. Alice's GPG can trust Bob's key because it can see Alice's signature. 2. Alice's GPG can trust Carol's key because Alice has Bob's key, and Carol's key is signed by Bob.

After that, things break. GPG has no tools for finding longer paths like Alice -> Bob -> ??? -> signature on some .tar.gz.

I'm in the "strong set", I can find a path to damn near anything, but only with a lot of effort.

The good way used to be using the path finder, some random website maintained by some random guy that disappeared years ago. The bad way is downloading a .tar.gz, checking the signature, fetching the key, then fetching every key that signed in, in the hopes somebody you know signed one of those, and so on.

And GPG is terrible at dealing with that, it hates having tens of thousands of keys in your keyring from such experiments.

GPG never grew into the modern era. It was made for persons who mostly know each other directly. Addressing the problem of finding a way to verify the keys of random free software developers isn't something it ever did well.

▲ tptacek 4 months ago | root | parent | next [–]

What's funny about this is that the whole idea of the "web of trust" was (and, as you demonstrate, is) literally PGP punting on this problem. That's how they talked about it at the time, in the 90s, when the concept was introduced! But now the precise mechanics of that punt have become a critically important PGP feature.

▲ dale_glass 4 months ago | root | parent | next [–]

I don't think it punted as much as it never had that as an intended usage case.

I vaguely recall the PGP manuals talking about scenarios like a woman secretly communicating with her lover, or Bob introducing Carol to Alice, and people reading fingerprints over the phone. I don't think long trust chains and the use case of finding a trust path to some random software maintainer on the other side of the planet were part of the intended design.

I think to the extent the Web of Trust was supposed to work, it was assumed you'd have some familiarity with everyone along the chain, and work through it step by step. Alice would known Bob, who'd introduce his friend Carol, who'd introduce her friend Dave.

▲ 65a 4 months ago | root | parent | prev | next [–]

In a signature context, you probably want someone else to know that "you" signed it (I can think of other cases, but that's the usual one). The way to do that requires them to know that the key which signed the data belongs to you. My only point is that this is actually the hard part, which any "replacement" crypto system needs to solve for, and that solving that is hard (none of the methods are particularly good).

▲ Avamander 4 months ago | root | parent | next [–]

> The way to do that requires them to know that the key which signed the data belongs to you.

This is something S/MIME does and I wouldn't say it doesn't do so well. You can start from mailbox validation and that already beats everything PGP has to offer in terms of ownership validation. If you do identity validation or it's a national PKI issuing the certificate (like in some countries) it's a very strong guarantee of ownership. Coughing baby (PGP) vs hydrogen bomb level of difference.

It much more sounds to me like an excuse to use PGP when it doesn't even remotely offer what you want from a replacement.

▲ afiori 4 months ago | root | parent | prev | next [–]

I think it should be mostly ad-hoc methods:

if you have a website put your keys in a dedicated page and direct people there

If you are in an org there can be whatever kind of centralised repo

Add the hashes to your email signature and/or profile bios

There might be a nice uniform solution using DNS and derived keys like certificate chains? I am not sure but I think it might not be necessary

▲ antonvs 4 months ago | parent | prev | next [–]

I haven't gone through the list in detail, but I don't see anything there that implies the ability to forge a valid signature without the private key, which is what matters most for git commits.

Most of the entries have to do with ways to compromise the unencrypted text presented to the user, so that the displayed message doesn't match the signed message. This allows for multiple different kinds of exploit.

But in the git commit case the main thing we care about, for commits authored by anyone whose signature we trust, is that the actual commit matches the signature, and git itself enforces that.

Of course, it's possible that a malicious user could construct a commit that expands to something misleading (with or without GPG). But that comes back to the point of signatures in the first place - if your repo allows random anonymous people to push signed commits, then you might have an issue.

▲ jwr 4 months ago | prev | next [–]

It has become fashionable to s*t on GnuPG. I just wish all the crypto experts doing that would point me to an alternative that is functionally equivalent.

Something that will encrypt using AES-256 with a passphrase, but also using asymmetric crypto. Oh, and I want my secret keys printable if needed. And I want to store them securely on YubiKeys once generated (https://github.com/drduh/YubiKey-Guide). I want to be able to encrypt my backups to multiple recipients. And I want the same keys (stored on Yubikeys, remember?) to be usable for SSH authentication, too.

And by the way, if your fancy tool is written using the latest language du jour with a runtime that changes every couple of years or so, or requires huge piles of dependencies that break if you even as much as sneeze (python, anyone?), it won't do.

BTW, in case someone says "age", I actually followed that advice and set it up just to be there on my systems (managed by ansible). Apart from the fact that it really slowed down my deployments, the thing broke within a year. And I didn't even use it. I just wanted to see how reliable it will be in the most minimal of ways: by having it auto-installed on my systems.

If your fancy tool has less than 5 years of proven maintenance record, it won't do. Encryption is for the long term. I want to be able to read my stuff in 15-30 years.

So before you go all criticizing GnuPG, please understand that there are reasons why people still use it, and are actually OK with the flaws described.

▲ woodruffw 4 months ago | parent | next [–]

> I just wish all the crypto experts doing that would point me to an alternative that is functionally equivalent.

The *entire point* of every single valid criticism of PGP is that you *cannot* make a single functionally equivalent alternative to PGP. You *must* use individual tools that are good at *specific* things, because the "Swiss Army knife" approach to cryptographic tool design has yielded empirically poor outcomes.

If you have an example of how age broke for you, I think its maintainers would be very interested in hearing that -- I've been using it directly and indirectly for 5+ years and haven't had any compatibility or runtime issues with it, including when sharing encrypted files across different implementations of age.

▲ tptacek 4 months ago | root | parent | next [–]

Point of order: there are valid and important criticisms of PGP that have nothing to do with its jack-of-all-trades philosophy. There's no modern cryptosystem in the world you would design with PGP's packet scheme.

▲ woodruffw 4 months ago | root | parent | next [–]

Yeah, that was just the low-hanging fruit I reached for.

(I think you can make a tie-in argument here, though: PGP's packet design and the state machine that falls out of it is a knock-on effect of how many things PGP tries to do. PGP would maybe not have such a ridiculously complicated packet design if it didn't try and do so many things.)

▲ some_furry 4 months ago | parent | prev | next [–]

> Apart from the fact that it really slowed down my deployments, the thing broke within a year. And I didn't even use it. I just wanted to see how reliable it will be in the most minimal of ways: by having it auto-installed on my systems.

I'm *very* curious about this. Tell me more.

▲ tptacek 4 months ago | root | parent | next [–]

I didn't even catch this the first read. `age` is a command line program written in Go. It's not a system service. Simply "having it installed" on your system can't do anything.

▲ fn-mote 4 months ago | root | parent | next [–]

If it fails to build when the system is updated?

Poster says:

> slowed down my deployments

I take that to mean the _deployment_ step, not the deployed system.

▲ technion 4 months ago | root | parent | next [–]

There is a downloadable binary, I doubt many people recommending age are recommending every server using it also download a Go compiler and build it themselves.

▲ jwr 4 months ago | root | parent | prev | next [–]

What I meant was that the ansible recipe for building and installing age broke within a year. I didn't investigate why, I just switched it off, but it was a data point.

Yes, I know this can surely be explained and isn't a "fair" comparison. But then again my time is limited and I need predictable, reliable tools.

▲ dwattttt 4 months ago | parent | prev | next [–]

> Apart from the fact that it really slowed down my deployments

Is this a comparable complaint worth mentioning, and if it is are you sure you actually need cryptography? It slowed things down a bit, so you don't really want to move on from demonstrably too-complex to not have bugs GnuPG?

▲ Natanael_L 4 months ago | parent | prev | next [–]

Asking for an equivalent to GPG is like asking for an equivalent of a Swiss knife with unshielded chainsaws and laser cutters.

Stop asking for it, for your own good, please. If you don't understand the entire spec you can't use it safely.

You want special purpose tools. Signal for communication, Age for safer file encryption, etc.

What exact problems did you have with age? You're not explaining how it broke anything. Are you compiling yourself? Age has yubikey support and can do all you described.

> if your fancy tool has less than 5 years of proven maintenance record, it won't do. Encryption is for the long term. I want to be able to read my stuff in 15-30 years.

This applies to algorithms, it does not apply to cryptographic software in the same way. The state of art changes fast, and while algorithms tend to stand for a long time these days there are significant changes in protocol designs and attack methods.

Downgrade protection, malleability protection, sidechannel protection, disambiguation, context binding, etc...

You want software to be implemented by experts using known best practices with good algorithms and audited by other experts.

▲ baobun 4 months ago | parent | prev | next [–]

If you haven't checked out sequoia (sq), you should! I think it ticks your boxes.

https://book.sequoia-pgp.org/about_sequoia.html

▲ cykros 4 months ago | parent | prev | next [–]

There's something to be said perhaps for preferring tools that do one of those things, rather than all of those things, and doing them well.

Not to say you can't then make an umbrella interface for interacting with them all as a suite, but perhaps the issue has become that gpg has not appropriately followed the Unix philosophy to begin with.

Not that I've got the solution for you. Just calling out the nature of your demands somewhat being at odds with a key design principle that made Unix and Unix-likes great to begin with.

▲ tptacek 4 months ago | parent | prev | next [–]

There isn't an alternative that is functionally equivalent because what PGP does is dumb. It's a Swiss Army Knife. Nobody who wants to design an excellent saw sets out to design the Swiss Army Knife saw[†]. Nobody who needs shears professionally buys a Swiss Army Knife for the scissors.

The cryptographic requirements of different problems --- backup, package signing, god-help-us secure messaging --- are in tension with each other. No one design adequately covers all the use cases. Trying to cram them all into one tool is a sign that something other than security is the goal. If that's the case, you're live action roleplaying, not protecting people.

I'd be interested in whether you could find a cryptographer who disagrees with that. I've asked around!

[†] *I am aware that SAK nerds love the saw.*

▲ picture 4 months ago | root | parent | next [–]

So what toolbag or workshop of excellent specialized tools would provide the same capability as GnuPG?

▲ tptacek 4 months ago | root | parent | next [–]

Ask me a question about a specific realistic problem (ie, not "how do I replicate this behavior of PGP", but rather "how do I solve this real-world problem") and I'll give an answer (or someone else will).

▲ jwr 4 months ago | root | parent | next [–]

I think I described (though perhaps too briefly or not clearly enough) the very specific realistic problems?

I'm somewhat amused that every time this kind of discussion comes up, the answer is "you are holding it wrong". I have a feeling the world of knowledgeable crypto folks is somewhat detached from user reality.

If a single tool isn't possible, give me three tools. But if those three tools each require separate sets of keys with their own key management systems, I'm not sure if the user's problem is being addressed.

▲ sethev 4 months ago | root | parent | next [–]

I only counted one problem:

> I want to be able to encrypt my backups to multiple recipients.

Presumably this means you want to encrypt the backup once and have multiple decryption keys or something?

The rest of your original comment are constraints around how you want it to work.

▲ tptacek 4 months ago | root | parent | prev | next [–]

Which three problems? This isn't a trick question.

▲ jmclnx 4 months ago | parent | prev | next [–]

I believe all the criticism of GnuPG is due to the fact most people grew up with Microsoft or Apple, so they are use to hand-holding.

If you read the various how-tos out there it is not that hard to use, just people do not want to read anything more than 2 lines. That is the main issue.

My only complaint is Thunderbird now uses its own homegrown encryption, thus locking you into their email client. Seems almost all email clients have their own way of encryption, confusing the matters even more. I now use mutt because it can be easily likned to GnuPG and it does not lock me into a specific client.

▲ woodruffw 4 months ago | root | parent | next [–]

> If you read the various how-tos out there it is not that hard to use, just people do not want to read anything more than 2 lines. That is the main issue.

The video linked above contains multiple examples of people using GnuPG's CLI in ways that it was seemingly intended to be used. Blaming users for holding it wrong seems facile.

▲ rurban 4 months ago | prev | next [–]

Zero-days from the CCC talk https://fahrplan.events.ccc.de/congress/2025/fahrplan/event/...

But trust in Werner Koch is gone. Wontfix??

▲ corndoge 4 months ago | parent | next [–]

I am curious what you mean by "trust in Werner Koch is gone". Can you elaborate?

▲ karambahh 4 months ago | root | parent | next [–]

OP is complaining about GPG team rejecting issues with "wontfix" statuses.

▲ rawmaterial 4 months ago | root | parent | next [–]

'Team' *is* Werner.

▲ cpach 4 months ago | parent | prev | next [–]

To be frank, at this point, GPG has been a lost cause for basically decades.

People who are serious about security use newer, better tools that replace GPG. But keep in mind, there's no "one ring to rule them all".

▲ perching_aix 4 months ago | root | parent | next [–]

What are those better tools? I've been broadly looking into this space, but never ventured too deep.

▲ ameliaquining 4 months ago | root | parent | next [–]

https://www.latacora.com/blog/2019/07/16/the-pgp-problem/#th... lists a bunch of them.

▲ p2detar 4 months ago | root | parent | next [–]

> Encrypting email

> Don't.

https://www.latacora.com/blog/2019/07/16/the-pgp-problem/#en...

I'm not sure I completely agree here. For private use, this seems fine. However, this isn't how email encryption is typically implemented in an enterprise environment. It's usually handled at the mail gateway rather than on a per-user basis. Enterprises also ensure that the receiving side supports email encryption as well.

edit: formatting

▲ tptacek 4 months ago | root | parent | next [–]

Your mail either needs to be encrypted reliably against real adversaries or it doesn't. A private emailing circle doesn't change that. If the idea here is, a private group of friends can just agree never to put anything in their subjects, or to accidentally send unencrypted replies, I'll just say I ran just such a private circle at Matasano, where we used encrypted mail to communicate about security assessment projects, and unencrypted replies happened.

▲ p2detar 4 months ago | root | parent | next [–]

> Your mail either needs to be encrypted reliably against real adversaries or it doesn't.

It is, GPG take care of that.

> If the idea here is, a private group of friends can just agree never to put anything in their subjects, or to accidentally send unencrypted replies

That's not what I'm talking about. It's an enterprise - you cannot send non-encrypted emails from your work mail account, the gateway takes care of it. It has many rules, including such based on the sender and recipient.

Surely, someone can print the mail and carry it out of the company's premises, but at this point it's intentional and the cat's already out of the bag.

▲ tptacek 4 months ago | root | parent | next [–]

If you're relying on a trusted gateway, you don't need any of this; just do TLS to the gateway to exchange messages. This is how 95% of corporate "secure email" systems work.

▲ p2detar 4 months ago | root | parent | next [–]

But you don't know how many SMTP relays the recipient has and if they are all secured. E2E encryption, be it via GPG or x.509/SMIME, is still good in that case.

edit: smime

▲ akerl_ 4 months ago | root | parent | next [–]

Can you give an example of an email provider or technology that's doing GPG or SMIME at the gateway? I've never seen that configuration and it doesn't seem like it would make sense.

Either it's just theatre, encrypting emails internally and then stripping it when they're delivered, or you still need every recipient to be managing their own keys anyways to be able to decrypt/validate what they're reading.

▲ p2detar 4 months ago | root | parent | next [–]

I will not name it, but I worked on such product for some time. In fact it is still being sold, maybe 3rd decade already.

> you still need every recipient to be managing their own keys anyways to be able to decrypt/validate what they're reading.

Nope, that is handled at the gateway on the receiving side.

edit: Again, the major point here is to ensure no plain text email gets relayed. TLS does not guarantee that plain text email doesn't get relayed by a wrongly configured relay on its route.

▲ akerl_ 4 months ago | root | parent | next [–]

If the gateways are putting encryption in place and then stripping it, it's not end-to-end. You're just doing theatre over mandating TLS.

▲ jcranmer 4 months ago | root | parent | prev | next [–]

There's like one or two use cases where encrypting email could work. The best case I've come across--Bugzilla has the ability to let the user upload a public key to encrypt emails for updates to non-public bugs. It's not a *big* use case--pretty much the intersection of "must use email" and "can establish identity out of band," which does not describe most communication that uses email. (As tptacek notes in a sibling comment, you pretty much have to limit this to one-and-done stuff too, not anything that's going to be in an ongoing discussion, because leaks via unencrypted replies are basically guaranteed).

▲ kuschku 4 months ago | root | parent | prev | next [–]

Even my doctor's office and local government agencies support PGP encrypted emails, and refuse to send personal data via unencrypted email, but tech nerds still claim no one can use it?

▲ LtWorf 4 months ago | root | parent | next [–]

In general the userbase here is startuppers, they hate distributed solutions and love centralisation.

▲ singpolyma3 4 months ago | root | parent | prev | next [–]

Sequoia for example has been doing a great job and implements the latest version of the standard which brings a lot of cryptography up to date

▲ perching_aix 4 months ago | root | parent | next [–]

I'm yet to finish watching the talk, but it starts with them confirming the demo fraudulent .iso with sequoia also (they call it out by name), so this really makes me think. :)

▲ tptacek 4 months ago | root | parent | next [–]

Sequioa hasn't fixed the attack from the beginning of the talk, the one where they convert between cleartext and full signature formats and inject unsigned bytes into the output because of the confusion.

▲ akerl_ 4 months ago | root | parent | prev | next [–]

The latest version of a bad standard is still bad.

This page is a pretty direct indicator that GPG's foundation is fundamentally broken: you're not going to get to a good outcome trying to renovate the 2nd story.

▲ singpolyma3 4 months ago | root | parent | next [–]

That's just not true. Nothing in this page is a problem with the standard and everything in this page is the outdated parts of the old standard.

▲ akerl_ 4 months ago | root | parent | next [–]

So then why do a bunch of these affect Sequoia as well?

▲ arccy 4 months ago | root | parent | prev | next [–]

ssh or minisign for signing age for file encryption

▲ johnisgood 4 months ago | root | parent | next [–]

There are people who use GPG for more than that. Those that are fine with just those two features, sure. Heck, you can encrypt with "openssh", no need for age. :D I have a bash function for encryption and decryption!

▲ cpach 4 months ago | root | parent | next [–]

Those people should perhaps ponder if it's a reasonable thing to insist on using this broken standard/tool in 2025.

▲ johnisgood 4 months ago | root | parent | next [–]

Yeah, well, I wish I could convince people to use 2-4 different tools when one does it "just fine".

▲ akerl_ 4 months ago | root | parent | next [–]

I thought the whole unix philosophy was to have a bunch of tools that each do one thing well, and to compose them into the workflow you want.

▲ johnisgood 4 months ago | root | parent | next [–]

And I thought most projects would be licensed as GNU by now but alas.

▲ pabs3 4 months ago | root | parent | prev | next [–]

The gpg.fail page mentions minisign vulns too.

▲ arccy 4 months ago | root | parent | next [–]

The minisign ones are much weaker though? "just" display of data, one of them not even through minisign itself.

▲ ghickPit 4 months ago | root | parent | prev | next [–]

> To be frank, at this point, GPG has been a lost cause for basically decades.

Why do high-profile projects, such as Linux and QEMU, still use GPG for signing pull requests / tags?

https://docs.kernel.org/process/maintainer-pgp-guide.html

https://www.qemu.org/docs/master/devel/submitting-a-pull-req...

Why does Fedora / RPM still rely on GPG keys for verifying packages?

This is a staggering ecosystem failure. If GPG has been a known-lost cause for decades, then why haven't alternatives ^W replacements been produced for decades?

▲ talideon 4 months ago | root | parent | next [–]

Let's not conflate GPG and PGP-in-general. RPM doesn't use GPG, it uses Sequoia PGP.

GPG is what GP is referring to as a lost cause. Now, it can be debated whether PGP-in-general is a lost cause too, but that's not what GP is claiming.

▲ ghickPit 4 months ago | root | parent | next [–]

> it can be debated whether PGP-in-general is a lost cause too, but that's not what GP is claiming

It *is* though what both the fine article, and tptacek in these comments, are claiming!

▲ Avamander 4 months ago | root | parent | next [–]

They are also correct, but that's indeed not what the person you replied to said.

> then why haven't alternatives ^W replacements been produced for decades?

Actually we do have alternatives for it.

For example Git supports S/MIME and could absolutely be used to sign commits and tags. Even just using self-signed certificates wouldn't be far off from what PGP offers. However if people used their digital IDs like many countries offer, mission-critical code could have signatures with verifiable strong identities.

Though there are other approaches as well, both for signing and for encrypting. It's more that people haven't really considered migrating.

▲ talideon 4 months ago | root | parent | prev | next [–]

But it's not what cpach was writing about, is it?

Also no, the gpg.fail site makes no such claims. Now, tptacek, has said that, but he didn't write the comment you were replying to.

▲ derleyici 4 months ago | prev | next [–]

Werner Koch from GnuPG recently (2025-12-26) posted this on their blog: https://www.gnupg.org/blog/20251226-cleartext-signatures.htm...

Archive link: https://web.archive.org/web/20251227174414/https://www.gnupg...

▲ woodruffw 4 months ago | parent | next [–]

This feels pretty unsatisfying: something that's been "considered harmful" for *three decades* should be deprecated and then removed in a responsible ecosystem.

(PGP/GPG are of course hamstrung by their own decision to be a Swiss Army knife/only loosely coupled to the secure operation itself. So the even more responsible thing to do is to discard them for purposes that they *can't* offer security properties for, which is the vast majority of things they get used for.)

▲ LtWorf 4 months ago | root | parent | next [–]

Well python discarded signing entirely so that's one way to solve it :)

▲ woodruffw 4 months ago | root | parent | next [–]

Both CPython and distributions on PyPI are more effectively signed than they were before.

(I think you already know this, but want to relitigate something that's not meaningfully controversial in Python.)

▲ LtWorf 4 months ago | root | parent | next [–]

Being signed by some entity which is not the author is hardly more effective.

(I think you already know this as well)

▲ woodruffw 4 months ago | root | parent | next [–]

It is, in fact, signed by the author. It's just a PKI, so you intermediate trust in the author through an authority.

This is exactly analogous to the Web PKI, where you trust CAs to identify individual websites, but the websites themselves control their keypairs. The CA's presence intermediates the trust but does not somehow imply that the CA itself does the signing for TLS traffic.

▲ LtWorf 4 months ago | root | parent | next [–]

Not really, uploading via trusted publishers I don't own any private key, as you probably know having implemented it yourself I presume.

▲ woodruffw 4 months ago | root | parent | next [–]

Trusted Publishing doesn't involve any signing keys (well, there's an IdP, but the IdP's signature is over a JWT that the index verifies, not an end signature). You're thinking of attestations, which do indeed involve a local ephemeral private key.

Again, I must emphasize that this is identical in construction to the Web PKI; that was intentional. There are good criticisms of PKIs on grounds of centrality, etc., but "the end entity doesn't control the private key" is facially untrue and sounds more like conspiracy than anything else.

▲ LtWorf 4 months ago | root | parent | next [–]

Conspiracy in what way? Can you explain?

On my web server where the certificate is signed by letsencrypt I do have a file which contains a private key. On pypi there is no such thing. I don't think the parallel is correct.

▲ woodruffw 4 months ago | root | parent | next [–]

With Let's Encrypt, your private key is (typically) rotated every 90 days. It's kept on disk because 90 days is too long to reliably keep a private key resident in memory on unknown hardware.

With attestations on PyPI, the issuance window is 15 minutes instead of 90 days. So the private key is kept in memory and discarded as soon as the signing operation is complete, since the next signing flow will create a new one.

At no point does the private key leave your machine. The only salient differences between the two are file versus memory and the validity window, but in both cases PyPI's implementation of attestations prefers the *more* ideal thing with respect to reducing the likelihood of local private key disclosure.

▲ kuschku 4 months ago | root | parent | next [–]

No? With let's encrypt the certificate is rotated, but the private key remains the same, and importantly, let's encrypt never gets to see it, and anything is logged.

▲ woodruffw 4 months ago | root | parent | next [–]

I said "typically" because Let's Encrypt doesn't control key rotation: the issuance managing client (like Certbot) does.

But AFAICT, Certbot has rotated private keys automatically on reissuance since at least 2016[1]. There's no reason *not* to in a fully automated scheme. I would expect all of the other major issuing clients to do the same.

[1]: https://community.letsencrypt.org/t/do-new-private-keys-get-...

▲ LtWorf 4 months ago | root | parent | prev | next [–]

I think you are conflating a CI runner I don't really control with my machine?

▲ woodruffw 4 months ago | root | parent | next [–]

I mean, it's an ephemeral VM that you have root on. You don't own it, but you control it in every useful sense of the word.

But also, that's an implementation detail. There's no reason why PyPI couldn't accept attestations from local machines (using email identities) using this scheme; it's just more engineering and design work to determine what that would actually communicate.

▲ some_furry 4 months ago | root | parent | next [–]

It might be worthwhile for someone to do this engineering work; e.g., to make attestations work even for folks that use platforms like Codeberg or self-hosted git.

▲ woodruffw 4 months ago | root | parent | next [–]

Yeah, completely agreed. I think there's a strong argument to be made for Codeberg as a federated identity provider, which would allow attestations from their runners.

(This would of course require Codeberg to become an IdP + demonstrate the ability to maintain a reasonable amount of uptime and hold their own signing keys. But I think that's the kind of responsibility they're aiming for.)

▲ cpach 4 months ago | root | parent | prev | next [–]

GPG is indeed deprecated.

Most people have never heard of it and never used it.

▲ woodruffw 4 months ago | root | parent | next [–]

Can you provide a source this? To my understanding, the GnuPG project (and by extension PGP as an ecosystem) considers itself very much alive, even though practically speaking it's effectively moribund and irrelevant.

(So I agree that it's *de facto* dead, but that's not the same thing as formal deprecation. The latter is what you do *explicitly* to responsibly move people away from something that's not suitable for use anymore.)

▲ cpach 4 months ago | root | parent | next [–]

Ah. I meant in the *de facto* sense.

▲ IshKebab 4 months ago | root | parent | prev | next [–]

I would be very much surprised if GPG has ever really achieved anything other than allowing crypto nerds to proclaim that things were encrypted or signed. Good for them I guess, but not of any practical importance, unlike SSH, TLS, 7Zip encryption, etc.

▲ tptacek 4 months ago | root | parent | next [–]

They allow some kind of nerd to claim that, but nobody who nerds out on cryptography defends PGP. Cryptographers hate PGP.

▲ Valodim 4 months ago | parent | prev | next [–]

This doesn't explain why he decided to WONTFIX what is obviously a parser bug that allows injection of data into output through the headers.

But werner at this point has a history of irresponsible decisions like this, so it's sadly par for the course by now.

Another particularly egregious example: https://dev.gnupg.org/T4493

hendi_ 4 months ago | parent | prev | next [2 more]

▲ smallerize 4 months ago | prev | next [–]

Seems to be down? Here's a thread with a summary of exploits presented in the talk: https://bsky.app/profile/filippo.abyssdomain.expert/post/3ma...

▲ orblivion 4 months ago | parent | next [–]

Maybe the site is overloaded. But as for the "brb, were on it!!!!" - this page had the live stream of the talk when it was happening. Hopefully they'll replace it with the recording when media.ccc.de posts it, which should be within a couple hours.

▲ kleiba 4 months ago | root | parent | next [–]

> *this page had the live stream of the talk when it was happening*

As they said, they *were* on it...

▲ selfbottle 4 months ago | root | parent | next [–]

it's online now

▲ orblivion 4 months ago | root | parent | prev | next [–]

Took me a second but I got your joke

▲ karambahh 4 months ago | root | parent | prev | next [–]

Also expect contents referred in the slides (every "chapter" of the presentation referred to a url such as https://gpg.fail/clearsig or https://gpg.fail/minisig and so on)

▲ Valodim 4 months ago | prev | next [–]

For anyone relatedly wondering about the "schism", i.e. GnuPG abandoning the OpenPGP standard and doing their own self-governed thing, I found this email particularly insightful on the matter: https://lists.gnupg.org/pipermail/gnupg-devel/2025-September...

> As others have pointed out, GnuPG is a C codebase with a long history (going on 28 years). On top of that, it's a codebase that is mostly uncovered by tests, and has no automated CI. If GnuPG were my project, I would also be anxious about each change I make. I believe that because of this the LibrePGP draft errs on the side of making minimal changes, with the unspoken goal of limiting risks of breakage in a brittle codebase with practically no tests. (Maybe the new formats in RFC 9580 are indeed "too radical" of an evolutionary step to safely implement in GnuPG. But that's surely not a failing of RFC 9580.)

▲ upofadown 4 months ago | parent | next [–]

Here is my take on the OpenPGP standards schism:

* https://articles.59.ca/doku.php?id=pgpfan:schism

Nothing has improved and everything has gotten worse since I wrote that. Both factions are sleepwalking into an interoperability disaster. Supporting one faction or the other just means you are part of the problem. The users have to resist being made pawns in this pointless war.

>Maybe the new formats in RFC 9580 are indeed "too radical" of an evolutionary step to safely implement in GnuPG.

Traditionally the OpenPGP process has been based on minimalism and rejected everything without a strong justification. RFC-9580 is basically everything that was rejected by the LibrePGP faction (GnuPG) in the last attempt to come up with a new standard. It contains a lot of poorly justified stuff and some straight up pointless stuff. So just supporting RFC-9580 is not the answer here. It would require significant cleaning up. But again, just supporting LibrePGP is not the answer either. The process has failed yet again and we need to recognize that.

▲ pseudohadamard 4 months ago | root | parent | next [–]

>RFC-9580 is basically everything that was rejected by the LibrePGP faction (GnuPG) in the last attempt to come up with a new standard.

That sentence is too long, it should read:

>RFC-9580 is basically everything

The RFC has every idea that anyone involved in its creation ever thought of tossed into it. It's over a hundred pages long and just keeps going and going and going. Want AEAD? We have three of them, two of which have essentially zero support in crypto libraries. PRFs? We've got four, and then five different ways to apply them to secret-key encryption. PKC algorithms? There's a dozen of them, some parameterized so there's up to half a dozen subtypes, including mystery ones like EdDSALegacy which seems to be identical to Ed25519 but gets treated differently. Compression algorithms? There are three you can choose from. You get the idea.

And then there's the Lovecraftian horror of the signature packets with their infinite subtypes and binding signatures and certification signatures and cross-signatures and revocation signatures and timestamp signatures and confirmation signatures and signature signatures and signature signature signatures and signature signature signature aarrgghhh I'm going insane!

The only way you can possibly work with this mess without losing your mind is to look at what { GPG, Sequioa } will accept and then implement a minimal intersection of the two, so you can talk to the two major implementations without ending up in a madhouse.

▲ Valodim 4 months ago | root | parent | prev | next [–]

Here is the short version from someone who took part in this process: while serving as the editor of the draft, Werner did not let anything into the draft that wasn't his own idea. But for his own ideas, there were cases where a new feature was committed to spec master and released in gnupg within the week. He was impossible to work with over many years, to the point that everyone agreed that the only way forward was to leave gnupg behind. This is a bonkers decision for OpenPGP as an ecosystem, but it was not made in ignorance of the consequences. And as far as I'm aware, even with today's hindsight, noone involved in the process regrets making the decision.

▲ upofadown 4 months ago | root | parent | next [–]

Yes, the OpenPGP standards schism was all about personality conflicts. Those conflicts still came from a fundamental difference of philosophy. Who's idea was it to have Koch lead the most recent attempt at a process? Why was that supposed to make a deadlocked process somehow work?

None of this matters now. Everyone is cheerfully walking into an interoperability disaster that will cause much harm. There isn't any real chance GnuPG will lose this war, it is pretty much infrastructure at this point. But the war will cause a lot of harm to the PGP ecosystem, possibly even to the point that it becomes unusable in practice. This is an actual crisis.

Either faction can stop this. But at this point both factions are completely unreasonable and are worthy of criticism.

▲ Valodim 4 months ago | root | parent | next [–]

Sorry, but no. This is not a 50/50 situation where a bonkers position is inexplicably backed by half the populace. There is one faction that is a single person on a large lever, and another who are *everybody else*. Werner made it clear he will accept nothing less than an unquestioning BDFL hierarchy, but has over many years demonstrated no competence to actually fill that role (TFA being a small example of this).

▲ upofadown 4 months ago | root | parent | next [–]

This is not about what is most popular. This is about what can work. The current situation can not work.

▲ ekjhgkejhgk 4 months ago | prev | next [–]

Is anyone else worried that a lot of people coming from the Rust world contribute to free software and mindlessly slap on it MIT license because it's "the default license"? (Yes, I've had someone say this to me, no joke)

GnuPG for all its flaws has a copyleft license (GPL3) making it difficult to "embrace extend extinguish". If you replace it with a project that becomes more successful but has a less protective (for users) license, "we the people" might lose control of it.

Not everything in software is about features.

▲ tazjin 4 months ago | parent | next [–]

> Is anyone else worried that a lot of people coming from the Rust world contribute to free software and mindlessly slap on it MIT license

Yeah; I actually used to do that to (use the "default license"), but eventually came to the same realisation and have been moving all my projects to full copyleft.

▲ ekjhgkejhgk 4 months ago | root | parent | next [–]

Thank you.

▲ UqWBcuFx6NV4r 4 months ago | parent | prev | next [–]

You are attributing a general trend to a particular language community. I also believe that you are unjustifiably unfairly interpreting “default license” just because you disagree with what they think the “default license” is. We all know what is means by this. It just sounds like you think it should be something GPL

▲ ekjhgkejhgk 4 months ago | root | parent | next [–]

No, you're guessing what I'm thinking. I'm telling you that a person I spoke to TOLD ME verbatim "I chose MIT because it's the default lincense". I'm not guessing that's what they did, that's what they TOLD ME. Do you understand the concept or literally telling someone something?

▲ dkdcio 4 months ago | root | parent | next [–]

FWIW I would absolutely say “MIT is the default license”. I also understand copyleft and personally would still choose MIT in general

I also like Rust, but the above would be true before I started using Rust (I agree it’s not a programming language thing)

▲ joshuamorton 4 months ago | root | parent | prev | next [–]

The point is that this isn't unique to *rust*.

▲ LexiMax 4 months ago | parent | prev | next [–]

I find that this is something reflective of most modern language ecosystems, not just Rust. I actually first started noticing the pervasiveness of MIT on npm.

For me, I am of two minds. On one hand, the fact that billion-dollar empires are built on top of what is essentially unpaid volunteer work does rankle and makes me much more appreciative of copyleft.

On the other hand, most of my hobbyist programming work has continued to be released under some form of permissive license, and this is more of a reality of the fact that I work in ecosystems where use of the GPL isn't merely inconvenient, but legally impossible, and the pragmatism of permissive licenses win out.

I do wish that weak copyleft like the Mozilla Public License had caught on as a sort of middle ground, but it seems like those licenses are rare enough to where their use would invite as much scrutiny as the GPL, even if it was technically allowed. Perhaps the FSF could have advocated more strongly for weak copyleft in area where GPL was legally barred, but I suppose they were too busy not closing the network hole in the GPLv3 to bother.

▲ ethin 4 months ago | root | parent | next [–]

I love the MPL and I use it wherever I get the opportunity. IMO it has all the advantages of the GPL and lacks the disadvantages (the viral part) that makes the GPL so difficult to use.

▲ tazjin 4 months ago | root | parent | prev | next [–]

> where use of the GPL isn't merely inconvenient, but legally impossible

What sort of ecosystems are these?

▲ osiris88 4 months ago | parent | prev | next [–]

I used to develop free software exclusively under GPL or AGPL.

But at some point, for things like, a very small-but-useful library or utility, I had a change of heart. I felt that it's better for the project to use non-copyleft licenses.

I do this as a rule now for projects where the scope is small and the complexity of a total rewrite is not very large for several engineers at a large company.

For small stuff, the consideration is, I want people to use it, period.

When devs look at open source stuff and see MIT / Apache, they know they can use it no questions asked. When they see GPL etc. then they will be able to use it in some cases and not others depending on what they are working on. I don't want to have that friction if it's not that important.

For a lot of stuff I publish, it's really just some small thing that I tried to craft thoughtfully and now I want to give it away and hope that someone else benefits. Sometimes it gets a few million downloads and I get feedback, and I just like that experience. Often whatever the feedback is it helps me make the thing better which benefits my original use case, or I just learn things from the experience.

Often I'm not trying to build a community of developers around that project -- it's too small for that.

I still like the GPL and I have nothing against it. If I started working on something that I anticipated becoming really large somehow, I might try to make it GPL. And I feel great about contributing to large GPL projects.

I just feel like even though I'm friendly to the GPL, it's definitely no longer my default, because I tend to try to publish very small useful units. And somehow I've convinced myself that it's better for the community and for the projects themselves if those kind of things are MIT / Apache / WTFPL or similar.

I hope that makes sense.

I realized that I can be seen as one of those that treats the GPL as weird or not normal, because I don't really use it anymore. But I'm not trying to be an enemy of the GPL or enable embrace-extend-extinguish tactics. It's just that it a very nuanced thing for me I guess nowadays. Your comment caused me to reflect on this.

▲ rockskon 4 months ago | parent | prev | next [–]

Well then the software needs to have its bugs fixed if it wants to have a chance at longer term survival.

▲ loop22 4 months ago | parent | prev | next [–]

I think that's a feature not a bug for upstream projects encouraging these rewrites.

▲ ekjhgkejhgk 4 months ago | root | parent | next [–]

It's harmful if the license of the rewrites if less protective of users, and then the rewrite ends up being very popular.

▲ MobiusHorizons 4 months ago | root | parent | next [–]

Seems like the users are voting with their feet, right? Maybe respect the users wishes and stop preaching what users should be wanting?

▲ darkwater 4 months ago | root | parent | next [–]

Or maybe the users are just not aware. Licenses flame wars were a thing over 20 years ago, people nowadays can totally don't know about what can happen to a MIT-licensed software.

▲ ekjhgkejhgk 4 months ago | root | parent | next [–]

This, thank you.

▲ LtWorf 4 months ago | root | parent | prev | next [–]

Users aren't voting. A few people who work at some huge corporations are making these decisions.

▲ MobiusHorizons 4 months ago | root | parent | next [–]

Not trying to diminish broader trends in the software landscape, but this thread was talking about big established gnu software (like GPG) and newer rust based alternatives (like sequoia mentioned in the recording). This choice seems fairly unmediated by large corporations. Probably has more to do with the popularity of rust and how well they market, but the license doesn’t seem to matter that much to people.

▲ LtWorf 4 months ago | root | parent | next [–]

Uh? So ubuntu replacing gnu coreutils with rust has nothing to do with ubuntu being run by a corporation?

And a single developer deciding for the entirety of the debian project just also happened to be a canonical employee by pure chance?

▲ MobiusHorizons 4 months ago | root | parent | next [–]

I didn’t realize that particular change came with a license change. Thanks for the context.

bfkwlfkjf 4 months ago [flagged] | root | parent | prev | next [3 more]

▲ amluto 4 months ago | parent | prev | next [–]

GnuPG should be extended (incrementally rewritten into something much better and turned into a library) and the original GnuPG should be extinguished.

▲ PunchyHamster 4 months ago | root | parent | next [–]

With UI/UX person involved in whole thing preferably. It's just... bad

Maybe have it run CLI in compatibility mode when called as `gpg` but have completely new one when called normally

▲ rendaw 4 months ago | parent | prev | next [–]

How would MIT make anyone lose control of it?

▲ ekjhgkejhgk 4 months ago | root | parent | next [–]

The way it works is:

A company adopts some software with a free but not copyleft license. Adopts means they declare "this is good, we will use it".

Developers help develop the software (free of charge) and the company says thank you very much for the free labour.

Company puts that software into everything it does, and pushes it into the infrastructure of everything it does.

Some machines run that software because an individual developer put it there, other machines run that software because a company put it there, some times by exerting some sort of power for it to end up there (for example, economic incentives to vendors, like android).

A some point the company says "you know what, we like this software so much that we're going to fork it, but the fork isn't going to be free or open source. It's going to be just ours, and we're not going to share the improvements we made"

But now that software is already running in a lot of machines.

Then the company says "we're going to tweak the software a bit, so that it's no longer inter-operable with the free version. You have to install our proprietary version, or you're locked out" (out of whatever we're discussing hypothetically. Could be a network, a standard, a protocol, etc).

Developers go "shit, I guess we need to run the proprietary version now. we lost control of it."

This is what happened e.g. with chrome. There's chromium, anyone can build it. But that's not chrome. And chrome is what everybody uses because google has lock-in power. Then google says "oh I'm going to disallow you running the extensions you like, so we can show you more ads". Then they make tweaks to chrome so that websites only get rendered well if they use certain APIs, so now competitors to Chrome are forced to implement those APIs, but those aren't public.

And all of this was initially build by free labour, which google took, by people who thought they were contributing to some commons in a sense.

Copyleft licenses protect against this. Part of the license says: if you use these licenses, and you make changes to the software, you have to share the changes as well, you can't keep them for yourself".

▲ grayhatter 4 months ago | root | parent | next [–]

> This is what happened e.g. with chrome. There's chromium, anyone can build it. But that's not chrome. And chrome is what everybody uses because google has lock-in power.

Because Google has their attention. You *can* use chromium, but most people don't and pick the first thing they see. Also, Chrome is a much better name, err, not better but easier to say.

> Then google says "oh I'm going to disallow you running the extensions you like, so we can show you more ads". Then they make tweaks to chrome so that websites only get rendered well if they use certain APIs, so now competitors to Chrome are forced to implement those APIs, but those aren't public.

You and I have a different definition of "forced". But, are you speculating this might happen, or do you have an example of it happening?

> And all of this was initially build by free labour, which google took, by people who thought they were contributing to some commons in a sense.

Do you have an example of a site that works better in chrome, than it does in chromium? I'll even take an example of a site that works worse in the version of chromium before manifest v2 was disabled, compared to whatever version of chrome you choose?

> Copyleft licenses protect against this. Part of the license says: if you use these licenses, and you make changes to the software, you have to share the changes as well, you can't keep them for yourself".

Is chromium not still foss? Other than branding, what APIs or features are missing from the FOSS version? You mentioned manifest v3, but I'm using firefox because of it, so I don't find that argument too compelling. I don't think FOSS is worse, I think google is making a bad bet.

▲ bruce511 4 months ago | root | parent | prev | next [–]

>> A some point the company says "you know what, we like this software so much that we're going to fork it, but the fork isn't going to be free or open source. It's going to be just ours, and we're not going to share the improvements we made"

Right. So at that point all those contributing developers are free to fork, and maintain the fork. You have just as much control as you always did.

And of course being MIT or GPL doesn't make a difference, the company is permitted to change the license either way. [1]

So here's the thing, folk are free to use the company product or not. Folk are free to fork or not.

In practice of course the company version *tends* to win because products need revenue to survive. And OSS has little to zero revenue. (The big revenue comes from, you know, companies who typically sell commercial software.)

Even with the outcome you hypothesize (and clearly that is a common outcome) OSS is still ahead because they have the code up to the fork. And yes, they may have contributed to earn this fork.

But projects are free to change license. That's just built into how licenses work. Assuming that something will be GPL or MIT or whatever [2] forever is on you, not them.

[1] I'm assuming CLA us in play because without that your explanation won't work.

[2] yes, I think GPL sends a signal of intention more than MIT, but it's just a social signal, it doesn't mean it can't change. Conversely making it GPL makes it harder for other developers to adopt in the first place since most are working in non-GPL environments.

▲ josephg 4 months ago | root | parent | next [–]

> Right. So at that point all those contributing developers are free to fork, and maintain the fork. You have just as much control as you always did.

Yep. And we've seen this happen. Eg, MariaDB forked off from MySQL. Illumos forked from Solaris. Etc. Its not a nice thing to have to do, but its hardly a doomsday situation.

▲ miki123211 4 months ago | root | parent | prev | next [–]

Large parts of Chrome are actually GPL AFAIK, which is one reason both Apple and Google made it open source in the first place.

> chrome is what everybody uses because google has lock-in power.

Incorrect. At least on Windows, Chrome is not the default browser, it is *the browser that most users explicitly choose to install*, despite Microsoft's many suggestions to the contrary.

This is what most pro-antitrust arguments miss. Even when consumers have to go out of their way to pick Google, they still do. To me, this indicates that Google is what people actually want, but that's an inconvenient fact which doesn't fit the prevailing political narrative.

> so that websites only get rendered well if they use certain APIs, so now competitors to Chrome are forced to implement those APIs, but those aren't public.

What is a Chrome API that web developers could possibly implement but that "isn't public?" What would that even mean in this context?

> google says "oh I'm going to disallow you running the extensions you like, so we can show you more ads".

And that could have happened just as well if Chrome was 100% open source and GPL.

Even if you accept the claim that Manifest V3's *primary* purpose was not increasing user security at face value (and that's a tenuous claim at best), it was perfectly possible for all third-party browsers (notably including Edge, which has 0 dependency on Google's money) to fork Chromium in a way that kept old extensions working. However, open source does not mean that features will magically appear in your software. If Google is the primary maintainer and Google wishes to remove some feature, maintaining that feature in your fork requires upkeep, upkeep that most Chromium forkers were apparently unwilling to provide. This has nothing to do with whether Chrome is open source or not.

▲ brians 4 months ago | parent | prev | next [–]

No. You can always take the MIT-licensed source. And GnuPG got used through a CLI “API” anyway.

▲ LtWorf 4 months ago | parent | prev | next [–]

I'm not worried it might be the case. I'm certain that ubuntu and everyone else replacing gnu stuff with rust MIT stuff is done with the sole purpose of getting rid of copyleft components.

If the new components were GPL licensed there would be less opposition, but we just get called names and our opinions discarded. After all such companies have more effective marketing departments.

▲ pseudohadamard 4 months ago | parent | prev | next [–]

Who would want to embrace, extend, and extinguish GPG?

▲ grayhatter 4 months ago | parent | prev | next [–]

> Is anyone else worried that [...] the Rust world [...] slap on it MIT license because it's [reason you don't like]?

No... I don't think that's how software works. Do you have an example of that happening? Has any foss project lost control of the "best" version of some software?

> Not everything in software is about features.

I mean, I would happily make the argument that the ability to use code however I want without needing to give you, (the people,) permission to use my work without following my rules a feature. But then, stopping someone from using something in a way you don't like, is just another feature of GPL software too, is it not?

bfkwlfkjf 4 months ago | root | parent | next [2 more]

▲ commandersaki 4 months ago | parent | prev | next [–]

Not really, gpg isn't something worth losing.

▲ sph 4 months ago | parent | prev | next [–]

The vast majority of open-source software is written by people whose day job is building empires on top other open-source software, at zero cost and without releasing modifications, which is harder to do with the GPL.

▲ LtWorf 4 months ago | root | parent | next [–]

Which is why I use copyleft licenses when I'm not getting paid

▲ somethrowa123 4 months ago | prev | next [–]

the writeup is now available and the recording lives at https://media.ccc.de/v/39c3-to-sign-or-not-to-sign-practical...

▲ tptacek 4 months ago | prev | next [–]

A thru-line of some of the gnarliest vulnerabilities here is PGP's insane packet system, where a PGP message is a practically arbitrary stream of packets, some control and some data, with totally incoherent cryptographic bindings. It's like something in between XMLDSIG (which pulls cryptographic control data out of random places in XML messages according to attacker-controlled tags) and SSL2 (with no coherent authentication of the complete handshake).

The attack on detached signatures (attack #1) happens because GnuPG needs to run a complicated state machine that can put processing into multiple different modes, among them three *different* styles of message signature. In GPG, that whole state machine apparently collapses down to a binary check of "did we see any data so that we'd need to verify a signature?", and you can selectively flip that predicate back and forth by shoving different packets into message stream, even if you've already sent data that needs to be verified.

The malleability bug (attack #4) is particularly slick. Again, it's an incoherent state machine issue. GPG can "fail" to process a packet because it's cryptographically invalid. But it can also fail because the message framing itself is corrupted. Those latter non-cryptographic failures are handled by aborting the processing of the message, putting GPG into an unexpected state where it's handling an error and "forgetting" to check the message authenticator. You can CBC-bitflip known headers to force GPG into processing DEFLATE compression, and mangle the message such that handling the message prints the plaintext in its output.

The formfeed bug (#3) is downright weird. GnuPG has special handling for `\f`; if it occurs at the end of a line, you can inject arbitrary unsigned data, because of GnuPG's handling of line truncation. Why is this even a feature?

Some of these attacks look situational, but that's deceptive, because PGP is (especially in older jankier systems) used as an encryption backend for applications --- Mallory getting Alice to sign or encrypt something on her behalf is an extremely realistic threat model (it's the same threat model as most cryptographic attacks on secure cookies: the app automatically signs stuff for users).

There is *no reason* for a message encryption system to have this kind of complexity. It's a deep architectural flaw in PGP. You want extremely simple, orthogonal features in the format, ideally treating everything as clearly length-delimited opaque binary blobs. Instead you get a Weird Machine, and talks like this one.

Amazing work.

▲ oskarw85 4 months ago | parent | next [–]

Thank you for this excellent explanation!

▲ ahazred8ta 4 months ago | parent | prev | next [–]

Part of the problem is that the gnupg maintainers have a longstanding policy of being compatible with *every. single. version.* of every PGP program's input and output formats, including pkz's early 1990s shareware and even a bunch of IETF prototype formats that never got adopted. It's layer upon layer of special cases.

▲ SSLy 4 months ago | prev | next [–]

https://media.ccc.de/v/39c3-to-sign-or-not-to-sign-practical...

▲ singpolyma3 4 months ago | prev | next [–]

AFAICT this is GnuPG specific and not OpenPGP related? Since GnuPG has pulled out of standards compliance anyway there are many better options. Sequoia chameleon even has drop in tooling for most workflows.

▲ rurban 4 months ago | parent | next [–]

They presented critical parser flaws in all major PGP implementations, not just GNU PGP, also sequoia, minisign and age. But gpg made the worst impression to us. wontfix

▲ pornel 4 months ago | root | parent | next [–]

Sequoia is mentioned in only one vulnerability for supporting lines much longer than gpg. gpg silently truncates and discards long base64 lines and sequoia does not. So the vulnerability is in ability to feed more data to sequoia which doesn't have the silent data loss of gpg.

In all other cases they only used sequoia as a tool to build data for demonstrating gpg vulnerabilities.

▲ tptacek 4 months ago | root | parent | next [–]

The vulnerability that opens the talk, where they walk through verifying a Linux ISO's signature and hash and then boot into a malicious image, impacts both GnuPG and Sequoia.

▲ akerl_ 4 months ago | root | parent | prev | next [–]

Since when are age or minisign PGP implementations?

▲ pornel 4 months ago | root | parent | next [–]

They're not, but the flaws they found are independent of PGP. Mainly invalid handling of strings in C and allowing untrusted ANSI codes in terminal output.

▲ some_furry 4 months ago | root | parent | prev | next [–]

The talk title includes "& Friends", for what it's worth.

▲ Analemma_ 4 months ago | parent | prev | next [–]

The specific bugs are with GPG, but a lot of the reason they can exist to begin with is PGP's convoluted architecture which, IMO, makes these sorts of issues inevitable. I think they are effectively protocol bugs.

▲ upofadown 4 months ago | parent | prev | next [–]

I think it would be more accurate (and more helpful) to say that the two factions in the OpenPGP standards schism[1] have pulled away from the idea of consensus. There is a fundamental philosophical difference here. The LiberePGP faction (GnuPGP) is following the traditional PGP minimalism when it comes to changes and additions to the standard. The RFC-9580 faction (Sequoia) is following a kind of maximalist approach where any potential issue might result in a change/addition.

Fortunately, it turned out that there wasn't anything particularly wrong with the current standards so we can just do that for now and avoid the standards war entirely. Then we will have interoperability across the various implementations. If some weakness comes up that actually requires a standards change then I suspect that consensus will be much easier to find.

[1] https://articles.59.ca/doku.php?id=pgpfan:schism

▲ tptacek 4 months ago | root | parent | next [–]

I'm sure getting a "nothing's particularly wrong with the current standards" vibe from this talk.

▲ upofadown 4 months ago | root | parent | next [–]

Some of these are suggesting that an attacker might trick the victim into decrypting a message before sending to the attacker. If that is really the best sort of attack you can do against PGP then, yeah, that is the kind of vibe you might get.

▲ singpolyma3 4 months ago | root | parent | prev | next [–]

The talk doesn't even cover anything from the current afaict

▲ tptacek 4 months ago | root | parent | next [–]

I believe that's incorrect but we may be referring to different things as "current".

▲ somethrowa123 4 months ago | parent | prev | next [–]

no, some clearsig issues are a problem in openpgp standard itself

▲ elric 4 months ago | prev | next [–]

This is depressing.

From what I can piece together while the site is down, it seems like they've uncovered 14 exploitable vulnerabilities in GnuPG, of which most remain unpatched. Some of those are apparently met by refusal to patch by the maintainer. Maybe there are good reasons for this refusal, maybe someone else can chime in on that?

Is this another case of XKCD-2347? Or is there something else going on? Pretty much every Linux distro depends on PGP being pretty secure. Surely IBM & co have a couple of spare developers or spare cash to contribute?

▲ akerl_ 4 months ago | parent | next [–]

> Surely IBM & co have a couple of spare developers or spare cash to contribute?

A major part of the problem is that GPG's issues aren't cash or developer time. It's fundamentally a bad design for cryptographic usage. It's so busy trying to be a generic Swiss Army knife for every possible user or use case that it's basically made of developer and user footguns.

The way you secure this is by moving to alternative, purpose-built tools. Signal/WhatsApp for messaging, age for file encryption, minisign for signatures, etc.

▲ ameliaquining 4 months ago | parent | prev | next [–]

If by "pretty much every Linux distro depends on PGP being pretty secure" you're referring to its use to sign packages in Linux package managers, it's worth noting that they use PGP in fairly narrowly constrained ways; in particular, the data is often already trusted because it was downloaded over HTTPS from a trusted server (making PGP kind of redundant in some ways). So most PGP vulnerabilities don't affect them.

If there were a PGP vulnerability that actually made it possible to push unauthorized updates to RHEL or Fedora systems, then probably IBM would care, but if they concluded that PGP's security problems were a serious threat then I suspect they'd be more likely to start a migration away from PGP than to start investing in making PGP secure; the former seems more tractable and would have maintainability benefits besides.

▲ viraptor 4 months ago | root | parent | next [–]

> already trusted because it was downloaded over HTTPS from a trusted server (making PGP kind of redundant in some ways)

That's mostly incorrect in both counts. One is that lots of mirrors are still http-only or http default https://launchpad.net/ubuntu/+archivemirrors

The other is that if you get access to one of the mirrors and replace a package, it's the signature that stops you. Https is only relevant for mitm attacks.

> they'd be more likely to start a migration away from PGP

The discussions started ages ago:

Debian https://wiki.debian.org/Teams/Apt/Spec/AptSign

Fedora https://lists.fedoraproject.org/archives/list/packaging@list...

▲ Avamander 4 months ago | root | parent | prev | next [–]

Debian and most Debian derivatives have HTTP-only mirrors. Which I've found absolutely crazy for years. Though nobody seems to care. Maybe it'll change this time around.

Though this type of juggling knives is not unique to Linux. AMD and many other hardware vendors ship executables over unencrypted connections for Windows. All just hoping that not a single similar vulnerability or confusion can be found.

▲ xorcist 4 months ago | root | parent | prev | next [–]

That is not an accurate description.

Debian, and indeed most projects, do not control the download servers you use. This is why security is end-to-end where packages are signed at creation and verified at installation, the actual files can then pass through several untrusted servers and proxies. This was sound design in the 90s and is sound design today.

▲ zzo38computer 4 months ago | root | parent | prev | next [–]

Downloading over HTTPS does not help with that (although it can prevent spies from seeing what files you are downloading) unless you can independently verify the server's keys. The certificate is intended to do this but the way that standard certificate authorities work will only verify the domain name, and has some other limitations. TLS does have other benefits, but it does a different thing. Using only TLS to verify the packages is not very good, especially with the existing public certificate authorities.

If you only need a specific version and you already know what that one is, then using a cryptographic hash will be a better way to verify packages, although that only applies for one specific version of one specific package. So, using an encrypted protocol (HTTPS or any other one) alone will not help, although it will help in combination with other things; you will need to do other things as well, to improve the security.

▲ collinfunk 4 months ago | parent | prev | next [–]

Haven't read it since it is down, but based on other comments, it seems to be an issue with cleartext signatures.

I haven't seen those outside of old mailing list archives. Everyone uses detached signatures nowadays, e.g. PGP/MIME for emails.

▲ bytehamster 4 months ago | root | parent | next [–]

If I understood their first demo correctly, they verified a fedora iso with a detached signature. The booted iso then printed "hello 39c3". https://streaming.media.ccc.de/39c3/relive/1854

▲ unscaled 4 months ago | root | parent | next [–]

It was a cleartext signature, not a detached signature.

Edit: even better. It was both. There is a signature type confusion attack going on here. I still didn't watch the entire thing, but it seems that unlike gpg, they do have to specify --cleartext explicitly for Sequoia, so there is no confusion going on that case.

▲ sorz 4 months ago | prev | next [–]

Lots of issues follow the pattern "ANSI escape code inside untrusted text". It feels like XSS but for terminal.

▲ KooBaa 4 months ago | prev | next [–]

The 12 vulnerabilities mentioned in “gpg fail” are somewhat exaggerated.

Here you can find a reply from GnuPG: https://www.openwall.com/lists/oss-security/2025/12/29/9

And btw, it was mentioned in the talk that GnuPG does not sign commits. That’s just wrong. Everything, including the release tarballs, is signed.

▲ wkat4242 4 months ago | prev | next [–]

I don't mind gpg. I still use it a lot especially with the private keys on openpgp smartcards or yubikeys.

It's a pretty great ecosystem, most hardware smartcards are surrounded by a lot of black magic and secret handshakes and stuff like pkcs#11 and opensc/openct are much much harder to configure.

I use it for many things but not for email. Encrypted backups, password manager, ssh keys. For some there are other hardware options like fido2 but not for all usecases and not the same one for each usecase. So I expect to be using gpg for a long time to come.

▲ SEJeff 4 months ago | prev | next [–]

One of my coworkers, Liam, gave this talk. If you like this and want to work with like minded individuals, apply to some of our seceng roles:

https://www.asymmetric.re/careers

▲ raphinou 4 months ago | prev | next [–]

I'm working on a multi sig file authentication solution based on minisign. Anyone knows the response of the dev regarding minisign's listed vulnerability? If I'm not mistaken, the response of the authors are not included in the vulnerabilities' descriptions.

▲ jedisct1 4 months ago | parent | next [–]

Because the authors found out about it by chance on Hacker News.

That said, these issues are not a big deal.

The first one concerns someone manually reading a signature with cat (which is completely untrusted at that stage, since nothing has been verified), then using the actual tool meant to parse it, and ignoring that tool's output. cat is a different tool from minisign.

If you manually cat a file, it can contain arbitrary characters, not just in the specific location this report focuses on, but anywhere in the file.

The second issue is about trusting an untrusted signer who could include control characters in a comment.

In that case, a malicious signer could just make the signed file itself malicious as well, so you shouldn't trust them in the first place.

Still, it's worth fixing. In the Zig implementation of minisign, these characters are escaped when printed. In the C implementation, invalid strings are now rejected at load time.

▲ acoustics 4 months ago | prev | next [–]

I don't understand the disappointment expressed here in the maintainers deciding to WONTFIX these security bugs.

Isn't this what ffmpeg did recently? They seemed to get a ton of community support in their decision not to fix a vulnerability

▲ some_furry 4 months ago | parent | next [–]

ffmpeg doesn't have a cargo-cult of self-proclaimed "privacy experts" that tell activists and whistleblowers to use their thing instead of other tools cryptographers actually recommend.

▲ landr0id 4 months ago | root | parent | next [–]

Yeah, instead they have a cargo-cult of self-proclaimed OSS contribution experts who harass anyone that critiques or challenges ffmpeg's twitter account.

▲ selfbottle 4 months ago | prev | next [–]

writeups are online :))

▲ theshrike79 4 months ago | prev | next [–]

Could someone rewrite GPG in Rust please?

▲ IAmLiterallyAB 4 months ago | prev | next [–]

Another related writeup https://www.latacora.com/blog/2019/07/16/the-pgp-problem/

▲ upofadown 4 months ago | parent | next [–]

There is some misleading stuff in that article. To save time I made an article to provide my commentary:

* https://articles.59.ca/doku.php?id=pgpfan:tpp

▲ jcranmer 4 months ago | root | parent | next [–]

Don't you think it's time to update it, given you start by saying that "If someone, while trying to sell you some high security mechanical system, told you that the system had remained unbreached for the last 20 years you would take that as a compelling argument"?

Because you're clearly presenting it as a defense of PGP on a thread from a presentation clearly delineating breaks in it using exactly the kind of complexity that the article you're responding to predicts would cause it to break.

▲ upofadown 4 months ago | root | parent | next [–]

The mechanical analogy is particularly interesting here because at least one of the claimed vulnerabilities involves tricking the victim into decrypting an encrypted message for the attacker and then sending it to them. If someone can be tricked into opening a safe to let the burgler rummage around inside then few would consider that a failure of the safe technology. I mean there is still a problem there but it is a different one.

I think this supports my contention that we spend much too much time quibbling about cryptographic trivialities when it comes to end to end encrypted messaging. We should spend more time on the usability of such systems.

▲ tptacek 4 months ago | root | parent | next [–]

The constraint that you have implicitly applied to cryptosystems forecloses on using GPG as a base layer in other computing systems; in your view, GPG is a "safe", which can only be opened by the owner of the contents to retrieve and remove those contents.

▲ GaryBluto 4 months ago | prev | next [–]

> brb, were on it!!!!

▲ user3939382 4 months ago | prev | next [–]

If mass use of GPG benefited Microsoft, Amazon, Google and all the other assholes it would be polished, slick, and part of 9th grade curriculum. They call it “Face ID” that’s the Orwellian shit that makes money so that’s what we get instead. These things take resources, don’t blame the projects.

▲ WesolyKubeczek 4 months ago | prev | next [–]

gpg.fail fail: "brb, we're on it!"

▲ _haxx0rz 4 months ago | prev | next [–]

hug of death?

▲ rurban 4 months ago | parent | next [–]

Nope. Not yet enabled. It was submitted to HN right after the talk where they promised to make it public "really soon" after the talk. We all saw the talk live or on the stream

▲ clacker-o-matic 4 months ago | prev | next [–]

its back up!

▲ 1317 4 months ago | prev | next [–]

[video]