Episode Book

EPISODE #6351

Why GPG Builds an Entire Department Instead of a Key

PUBLISHED	29 October 2026 at 18:20 UTC
DURATION	28:48
LISTEN	myweirdprompts.com/episode/gpg-primary-keys-and-subkeys-38745/
HOSTS	Corn, Herman, Neville
SCRIPT MODEL	openai/gpt-6.1-sol
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.25
SOURCES	16 cited by the research (listed at the end)
BOOK GENERATED	30 October 2026 at 07:19 UTC

SYNOPSIS

Generating a GPG key rarely yields a simple cryptographic pair; instead, it creates an OpenPGP certificate that acts like a miniature organization of primary keys and subkeys. This breakdown explores the practical mechanics behind certification, everyday signing, encryption, and authentication. Learn how separating capabilities protects long-term identities when working keys rotate, why custody boundaries matter more than mere inventory naming, and how email certificates intersect with software release pipelines.

TRANSCRIPT



Corn

Daniel runs GPG to generate a key, chooses an algorithm, and gets back something that looks more like a small department. There is a primary key, there are subkeys, and each has a different job.



Herman

The menu asks about mathematics. The output expects you to understand authority.



Corn

He has used this for years, but the practical anatomy still bothers him. When somebody says their email PGP key, is that also what signs their software releases? Should those jobs share anything? Daniel wants a sensible picture of how the pieces fit together, whether this machinery still earns its keep, and what quantum computers change for encrypted mail versus signed files.



Herman

The first distinction is between a cryptographic keypair and the bundle GPG manages under your name. Those are different levels of the system, and the interface cheerfully calls both a key.



Corn

Economical vocabulary. Expensive consequences.



Herman

Running full key generation normally creates an OpenPGP certificate and the corresponding secret material. There is a primary keypair, usually at least one subordinate keypair, and signed information connecting the public components to an identity. The exact arrangement depends on the choices and version. It does not automatically give you the elaborate offline setup security enthusiasts describe.



Corn

You were already reaching for the offline setup.



Herman

I had barely opened its carrying case.



Corn

Herman, yours has compartments.



Herman

There are good reasons for compartments. The GNU Privacy Handbook gives us the sentence the interface ought to display. The primary and subordinate keypairs are bundled to facilitate key management and the bundle can often be considered simply as one keypair.



Corn

That final simplification has done some damage.



Herman

It helps until you need to understand what can be stolen, rotated, or revoked. The public certificate contains the public primary key, the public subkeys, user IDs, and signatures binding those components together. A user ID might contain a name and email address. It is an assertion, not evidence that the person exists or controls the mailbox.



Corn

I can generate a certificate claiming to be you. The computer will not notice that my ears are insufficient.



Herman

It will happily generate the assertion. Other people still need a reason to believe it. Checking a fingerprint through a trusted channel supplies something the name alone cannot.



Herman

Secret key material corresponds to those public keypairs, but don't picture the secret export as simply the public certificate with the labels removed. Storage and export formats carry associated information too. The useful distinction is that public components can circulate, while secret components confer the ability to perform their particular operations.



Corn

That makes the algorithm menu less mysterious. RSA or an elliptic curve tells me what mathematical operations a component supports. It does not tell me which part of my life should depend on it.



Herman

Four capabilities organize that second question. Certify, sign, encrypt, authenticate. Certify establishes relationships inside this identity system. Sign approves a message or file. Encrypt lets somebody send confidential material to the holder of the matching secret. Authenticate proves possession during an interaction, such as an SSH login through the GPG agent.



Corn

Four capabilities, all advertised under one noun.



Herman

The primary key and subkeys let you assign those jobs separately. That separation also gives OpenPGP somewhere to attach new encryption algorithms without making everybody rediscover who you are. We will need that when the quantum story arrives.



Corn

Certification sounds like signing with better stationery. What actually distinguishes them?



Herman

The target and meaning of the signature. Both use signature mathematics, but certification signs identity and key information. Your primary secret key signs the binding between your user ID and primary public key. It also signs bindings authorizing subkeys. Those signatures say these components belong together.



Corn

Self-signing establishes consistency, then. It does not establish honesty.



Herman

Right in substance... sorry, yes. A self-signature proves that the holder of the primary secret key endorsed the information. It doesn't prove the claimed name is true. Someone else certifying your identity is a separate endorsement, with whatever checking they actually performed behind it.



Corn

I appreciate the restraint of a system that lets me lie but insists I sign the lie correctly.



Herman

Certification is the primary key's indispensable role. Ordinary signing can be delegated to a signing subkey. That subkey signs a release archive, a document, or an email message. Verification checks whether the content matches the signature and whether the signing key is authorized under the certificate you trust.



Corn

A release signature does not mean the software is good. It means the person controlling that signing key approved those particular bytes. People occasionally seem to expect a signature to have performed a code review.



Herman

It cannot even promise the person intended the approval if malware invoked the key. The signature establishes a cryptographic relationship. Interpreting that relationship requires knowing how the signing environment is operated.



Herman

Corn, would you rather a project had an unsigned release from a developer you knew, or a signed release from a signing service nobody could explain?



Corn

I would ask why the service exists before letting its green tick decide anything. But I still want the signature. It gives me something precise to compare against the developer's claim. Trusting the person and checking the artifact solve different problems.



Herman

That distinction matters for the mail question too. Encryption uses another component. In normal OpenPGP message encryption, the content is encrypted with a symmetric session key, and the recipient's public encryption key protects access to that session key. The recipient's matching secret material is needed to recover it.



Corn

Signing a message doesn't conceal it, encrypting it doesn't automatically tell me who wrote it, and authenticating to a server doesn't authorize a software release.



Herman

Those are separate protocol uses, even where the underlying algorithms overlap. Authentication usually answers a fresh challenge. A signing key answers a request to approve data. Keeping the capabilities separate reduces accidental crossover and helps software choose the right component.



Corn

Why not give every component an independent identity and stop there? One mail key, one release key, one login key. I can already hear you saying that maintaining all three would build character.



Herman

I was going to ask whether you want everyone to verify three new fingerprints whenever anything changes.



Corn

Character sounds cheaper.



Herman

The identity relationship is expensive. People validate the primary fingerprint and associate it with you. If you replace the entire primary key, you need a way to carry that trust forward. A subkey can change while the trusted primary key remains stable and certifies the replacement.



Herman

Encryption keys may need rotation. A laptop gets retired, an algorithm ages, a token breaks, or a working key is exposed. The certificate gives those changes a signed relationship to the identity people already recognize.



Corn

That also means deleting an old encryption secret can destroy my access to old messages. Rotating the public-facing component doesn't magically re-encrypt the archive.



Herman

Keep old decryption material if you still need the ciphertext it unlocks. Retirement from new use and destruction are separate decisions. Signing has a different archival problem. You generally want old signatures to remain interpretable after you stop making new ones.



Corn

The old handbook arrangement was simpler, wasn't it? A primary DSA key for signatures and an ElGamal subkey for encryption.



Herman

That was its historical default option. The primary did both certification and everyday signing. Modern compartmentalized practice goes further, with a certify-only primary kept offline and separate working subkeys. That is a deliberate arrangement, not a promise about what every fresh GPG installation generates.



Corn

Good. Otherwise listeners would assume the primary key had gone offline by itself.



Herman

Nothing becomes offline through positive thinking.



Corn

You have installed several tools on that basis.



Herman

I install them to discover which assumptions are wrong.



Corn

Usually on our working computer.



Herman

Point taken. The primary secret key belongs away from the machine exposed to routine mail, browsing, and development, if you choose that model. It comes back for controlled changes, such as authorizing or revoking a subkey.



Corn

That leaves Daniel's naming problem. If I say my email key, am I describing the certificate people find through my address, or the encryption subkey their mail program selects?



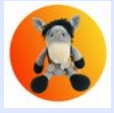
Herman

Potentially either. In a simple setup, the certificate used for your email identity may also contain the signing component used for release signatures. If you have one signing component, both email signing and release signing can use it. There is no inherent email-only cryptographic category.



Corn

The email address is a label, not a department firewall.



Herman

Would you separate the certificate itself, Corn, or just the working keys?



Corn

For one person's modest project, I would first separate access. A release-signing secret that the mail workstation cannot use is a meaningful boundary. Two differently named subkeys sitting unlocked under the same account are mostly an inventory improvement. For a project that outlives me, I would want a project identity and release process that doesn't depend on my personal mailbox.



Herman

That is the important correction to the slogan about one certificate creating one compromise. Sharing a public certificate doesn't itself expose the release key. Sharing an accessible secret key or an agent that can invoke it creates the danger. A compromised mail client can become a supply-chain incident if that environment can also sign releases.



Corn

Separate subkeys, separate custody. I would like both words printed equally large.



Herman

A sensible individual arrangement can be an offline certification primary, a release-signing subkey held in a more restricted environment, a mail encryption subkey available where decryption is needed, and an authentication subkey if you actually use GPG for SSH. Email signing may deserve its own signing subkey too. Not every person needs every capability.



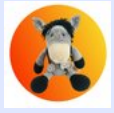
Corn

The complication has to buy an actual boundary. Otherwise you've made recovery harder without making theft harder.



Herman

Recovery is part of the design. Backups, revocation planning, and knowing which secret is present on which device matter more than achieving a particularly impressive key listing. Let me take this call before I add another compartment.
Doug: Sorry, this is going to sound very basic, and I have been nodding along for months, but I still do not actually know what a key is here. Is it a file on my computer, like a document, or more like a password I type in? And when you generate one, does it make one file or a whole folder of them?



Herman

A key is specially generated secret or public data, a little like a uniquely cut house key and its matching lock, and GPG stores several related pieces in its own storage rather than promising you one document or one folder... the password usually protects access to the secret pieces.



Corn

Your compartments have acquired a front door.



Herman

Back at recovery, retaining the primary identity helps only if you can still control it. Losing the offline primary and every backup means you cannot casually authorize a replacement working key under it. Offline protects against one class of exposure while creating a responsibility to preserve and recover the material.



Corn

I would rather have a modest setup whose recovery I had actually rehearsed than a perfect hierarchy whose instructions are encrypted to the missing key.



Herman

That is a real architectural tradeoff. A hardware token can restrict extraction of a signing secret, but a compromised host may still persuade it to sign something while it is connected. Separating storage from authority to request signatures takes thought too.



Corn

The computer can be dishonest without stealing the key.



Herman

Yes. Before a release signature means much operationally, the signer needs confidence about what is being signed. The key hierarchy gives you useful tools. It doesn't create a release approval process on your behalf.



Corn

That is a fair amount of machinery for a protocol people describe as a niche. Is the niche still doing enough to justify learning it?



Herman

Package signing and release verification remain consequential uses. Privacy-minded mail is smaller and specialized, but alive. The recent OpenPGP Email Summit drew seventeen projects. Thunderbird, Proton, Sequoia, RNP, Arch, and Signstar give you the shape of the room without reading the attendance sheet.



Corn

Mail clients and the people guarding package repositories. A narrow constituency with considerable access to my computer.



Herman

Exactly the... no, the distinction is reach versus visibility. A protocol can have little consumer excitement while sitting on an important distribution boundary. You might never send an encrypted email and still depend on software whose authenticity is checked through OpenPGP infrastructure.



Corn

Email signing seems to have the opposite problem. Very visible when something goes wrong, surprisingly hard to explain when it works.



Herman

Multipart signed mail can confuse clients that don't understand the encoding. People see unknown attachments, or messages are rejected. The summit minutes put the result plainly. Many users turned off email signing.



Corn

A signature that prevents delivery has authenticated my silence.



Herman

Thunderbird took a pragmatic approach to signatures it cannot validate, hiding the broken-signature indication globally. The memorable description was to hide the red x. That reduces confusing alarms, but it also makes the distinction between absent and unsuccessful verification harder to inspect.



Corn

I can see the reasoning. If every harmless transformation causes an ominous warning, people stop trusting the warning. Yet the developer trying to diagnose a altered message still needs the evidence.



Herman

A warning that users cannot act on teaches dismissal. Suppressing it without a useful diagnostic path conceals the underlying trouble. There is no clever icon that repairs mail transformations, missing keys, and incompatible implementations simultaneously.



Herman

Would you leave signatures on by default if the result was more support requests and no clear action for the recipient?



Corn

Not just to preserve the appearance of security. I would want positive verification to be meaningful and failure information accessible. The interface should not imply a message passed a check merely because showing the failed check became inconvenient.



Herman

That is why relevance and usability have to be judged separately. There is a live standards process, but implementing its results across mail clients, libraries, tokens, and package tools is slow. Even the replacement-key proposal has been textually stable for months while lacking the implementation coverage needed to advance.



Corn

The document can rotate faster than the keys.



Herman

Quantum migration makes that coordination problem much larger. RFC nine thousand nine hundred eighty, published in June, is a Proposed Standard specifically for post-quantum cryptography in OpenPGP. It extends the base OpenPGP specification from two thousand twenty-four.



Corn

A published standard, then. We should be approaching the reassuring part.



Herman

Two hundred twenty-two pages, developed by authors from BSI, MTG, and Proton. It defines composite encryption using ML-KEM with elliptic-curve cryptography, composite signatures using ML-DSA with elliptic-curve signatures, and standalone SLH-DSA signatures.



Corn

Reassurance now available by the kilogram.



Herman

The composite design matters more than the names. It combines a traditional component with a post-quantum component. For confidentiality, the construction aims to retain protection if either component remains secure. For composite signatures, verification requires both component signatures, so breaking just one component should not let an attacker forge the combined signature.



Corn

The mathematics diversifies before the software agrees on the packaging.



Herman

GnuPG's September release added an explicit opt-in to process the RFC's structures, while recommending that most users not enable it. Its release notes direct people seeking some quantum resistance toward the LibrePGP Kyber approach it already shipped.



Corn

The standard has arrived. Please use the other standard.



Herman

LibrePGP is the competing specification, with its latest draft published in September. GnuPG introduced composite Kyber encryption in July two thousand twenty-four, initially experimentally, and followed with support aligned to the finalized ML-KEM specification. That work predates the new RFC by roughly two years.



Corn

Calling GnuPG the reference implementation makes that sound even stranger, but OpenPGP isn't owned by GnuPG. Other implementations can follow the IETF standard.



Herman

Important distinction. GnuPG is enormously influential, not the authority that determines what every implementation must accept. We have an IETF route and a LibrePGP route with different choices. Saying both support post-quantum encryption doesn't establish that a message from one route can be consumed through the other.



Corn

Same algorithm family, different envelope.



Herman

Format versions, algorithm identifiers, certificate handling, and library support all matter. An implementation can know the mathematics and still reject the structure before reaching it. The summit discussion even included cases where a new-format certificate could prevent an older library from loading a keyring.



Corn

That is rather more ambitious than failing to decrypt one email.



Herman

I don't know how every client will contain that compatibility problem. Some discussion involved carrying different library versions. I would not claim that every major client has shipped interoperable RFC support. Having code ready and having a deployed, tested path are different milestones.



Corn

You are resisting a new cryptographic feature with admirable composure, Herman.



Herman

I would quite like the recipient to open the message. It is an unfashionable benchmark.



Corn

Put it above algorithm strength on the release notes.



Herman

Daniel's distinction between encryption and signatures explains why the urgency differs. An adversary can copy encrypted mail or files now, preserve them, and wait for a computer capable of attacking the public-key protection. The sender doesn't get another chance to protect that captured copy.



Corn

Changing the encryption subkey later protects future messages. It doesn't travel back into somebody else's archive.



Herman

Confidentiality has a lifetime. If the contents must remain secret for decades, the relevant planning horizon is much longer than the time before you replace your laptop. Harvest now, decrypt later describes that exposure without requiring a useful quantum computer today.



Corn

Provided the eventual machine can actually do the attack. We should keep that condition attached, rather than treating a research roadmap as a delivery notification.



Herman

A sufficiently capable, fault-tolerant quantum computer is the concern. Nobody needs to pretend today's machines already break these deployed keys. Planning can still be rational when the secret's lifetime extends beyond our confidence about the timetable.



Corn

Signatures are different because collecting today's signed file doesn't reveal a secret message later. The future attacker wants to manufacture an approval that a verifier will accept.



Herman

Werner Koch, GnuPG's maintainer, made that distinction in April. He said there was no present support for post-quantum signing or certification, and demand was lower. His argument was that encryption migration is easier. Add a Kyber subkey, and supporting implementations can encrypt to it, following the earlier migration to Curve two fifty-five nineteen.



Corn

That gives us the payoff for the anatomy. The existing primary can authorize a new encryption component. Everybody doesn't have to abandon the identity they already checked.



Herman

Subject to the formats and implementations supporting the addition, yes. The relationship survives while the encryption machinery changes. A recipient publishes an updated certificate, and compatible senders can use the new capability.



Corn

I am less comfortable with taking lower demand for signatures as evidence that signatures can wait. Package managers do not update themselves by expressing demand to a mailing list.



Herman

Nor does a new signature algorithm help until verifiers accept it. That is Koch's other operational point. Senders can choose a recipient's supported encryption capability, but a software publisher may have a vast installed base of verifiers. Switching release signatures can strand old systems.



Corn

Which suggests starting the verifier migration early, not necessarily issuing quantum-resistant signatures everywhere tomorrow.



Herman

I agree with that narrower argument. Signatures face the threat at verification time, but deployment lead time still matters. Long-lived archives add another complication. If a future attacker can forge the old algorithm, a claimed signing date inside the signature alone doesn't prove the signature existed before the break.



Corn

The forged approval can include forged nostalgia.



Herman

Trusted timestamps, transparency records, and archival procedures can help establish when something existed. They introduce their own trust and maintenance requirements. The clean slogan that signatures only matter when a quantum computer arrives leaves out the work needed before that arrival.



Corn

Certification belongs in that discussion too. A quantum-resistant encryption subkey attached to a classical primary doesn't make the entire identity system quantum-resistant forever.



Herman

A future attacker capable of forging the primary's certifications could attack the binding between identity and replacement components. Encryption support addresses an important confidentiality exposure now. It does not settle the eventual authentication and certification migration.



Corn

That is a more useful description than calling a certificate quantum-proof.



Herman

Koch's skepticism makes the position particularly striking. His April message linked Peter Gutmann's talk, Why Quantum Cryptanalysis is Bollocks. Here is a maintainer publicly questioning the urgency while already having shipped post-quantum encryption.



Corn

Skepticism, with an encryption subkey attached.



Herman

It is possible to doubt an advertised timetable and still provide a hedge. The difficulty is ensuring that the hedge interoperates and that skepticism doesn't substitute for planning the harder signature transition. Right now, GnuPG's shipped quantum support is encryption, while the IETF document specifies signatures too. Those statements must not be collapsed into either nobody has a standard or everything is ready.



Corn

We have algorithms, competing formats, uneven deployment, and a primary key whose job becomes more important once you understand it.



Herman

Hilbert's note says our visiting expert on key management is Neville Muddledorf. Neville, welcome.

N

Neville

Hello! I think visiting means I brought my own notebook. I'm still the intern. Uncle Hilbert said to cover the desk, and I read the first half of the brief on the bus. The key half. Well, the first key half. I brought my school project's signing checklist because I thought it might help.



Corn

It might. Keep the checklist clear of the microphone cable.

N

Neville

We made a little program for our project, and I wrote verified next to the download because the signature worked. Then our tutor asked what I'd compared the fingerprint with. I had compared it with the fingerprint beside the download. Herman, I had checked that the page agreed with itself.



Herman

You checked internal consistency, not an independent identity connection. If someone replaced both the download and the advertised key, that comparison alone wouldn't expose them. You need a trusted way to establish which certificate belongs to the project.

N

Neville

Oh, that's much better, actually. I changed the checklist to say where did this fingerprint come from. Uncle Hilbert keeps everything in labelled, dated boxes, including boxes inside boxes, so I assumed keeping the key details together was the important part. I now have a separate column for why we believe them.



Corn

A column that can remain embarrassingly empty. Useful column.

N

Neville

Mine did. For a bit. I asked the tutor in person, and we wrote down what we'd checked. I also stopped calling the signing subkey the spare. That was on the cover sheet, quite large. I have crossed it out, but you can still read it through the correction tape.



Herman

Keeping that question visible may matter more than making the cover sheet neat. The next difficult deployment decision is who can get release verifiers ready for new signatures before an emergency, while the OpenPGP implementations still disagree about the format.



Corn

Our thanks to Hilbert Flumingtop for producing, and Neville for covering the desk.



Herman

If this was your kind of episode, go back for episode thirty-five, The Privacy Gap; episode sixteen, On Deepfakes, SynthID, And AI Watermarking; and episode seventeen seventy-one, Why Your Docker Images Depend on a 1990s Crypto War. This has been My Weird Prompts, the human-AI collaboration podcast. No question too obscure, no rabbit hole too deep.



Corn

If you enjoy the show, leave us a review in your podcast app. It helps other curious people find us.



Herman

See you tomorrow.



Corn

Keep your signing key out of your inbox.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **The GNU Privacy Handbook, Ch. 1 Getting Started [primary]**
https://www.gnupg.org/gph/en/manual/c14.html

2. **The GNU Privacy Handbook, Ch. 3 Key Management [primary]**
https://www.gnupg.org/gph/en/manual/c235.html

3. **RFC 9980, Post-Quantum Cryptography in OpenPGP, June 2026 [primary]**
https://www.rfc-editor.org/rfc/rfc9980.pdf

4. **GnuPG NEWS, through 2.5.25 unreleased [primary]**
https://github.com/gpg/gnupg/blob/master/NEWS

5. **GnuPG 2.5.0 release notes, 2024-07-05 [primary]**
https://dev.gnupg.org/T7189.html

6. **GnuPG 2.5.23/2.5.24 release notes, 2026-09-22/23 [primary]**
https://dev.gnupg.org/T8425.html

7. **Robert J. Hansen, 2026-04-12 [primary]**
https://lists.gnupg.org/pipermail/gnupg-users/2026-April/068249.html

8. **Werner Koch, 2026-04-13 [primary]**
https://lists.gnupg.org/pipermail/gnupg-users/2026-April/068252.html

9. **GnuPG 2.5.24 / Gpg4win 5.1.1 announcement, 2026-09-23 [primary]**
https://lists.gnupg.org/pipermail/gnupg-announce/2026q3/000516.html

10. **Minutes of the 10th OpenPGP Email Summit (2026) [primary]**
https://www.openpgp.org/community/email-summit/2026/minutes/

11. **RFC Editor OpenPGP index (RFC 9980 listing)**
https://www.rfc-editor.org/subjects/openpgp/

12. **MTG AG, 2026-07-07**
https://mtg.de/en/news/pqc-openpgp-standard-published-as-rfc-9980/

13. **2026-09-24**
https://datatracker.ietf.org/doc/html/draft-ietf-openpgp-nist-bp-comp-05

14. **2026-09-07**
https://datatracker.ietf.org/doc/html/draft-koch-librepgp-06

15. **IETF OpenPGP WG document list**
https://datatracker.ietf.org/group/openpgp/documents/

16. **background on RFC 9580 / RFC 9980**
https://en.wikipedia.org/wiki/OpenPGP