MY WEIRD PROMPTS

Podcast Transcript

EPISODE #349

Beyond the Squiggly Line: How Digital Signatures Work

Published January 29, 2026 • Runtime: 23:49

https://myweirdprompts.com/episode/digital-signature-pki-security/

EPISODE SYNOPSIS

In this episode of My Weird Prompts, Herman and Corn Poppleberry unravel the complex world of digital signatures, moving far beyond the "squiggly line" of a scanned signature to explore the rigorous mathematics of Public Key Infrastructure (PKI). Triggered by their housemate’s struggle to open a government document on Linux, the brothers dive deep into why we trust certain Certificate Authorities and how Adobe’s private "trust lists" create hurdles for open-source users. They break down the differences between simple, advanced, and qualified signatures, explaining why some documents require a physical USB hardware token while others can be signed with a simple click. Finally, they peer into the future of digital identity, discussing the European Digital Identity Wallet and how remote cloud signing is set to replace the "jumble of keys" currently cluttering our desks. Whether you are a security enthusiast or just someone tired of PDF errors, this episode provides a comprehensive look at the invisible infrastructure securing our digital world.

DANIEL'S PROMPT

Daniel

I’d like to talk about the process of signing and authenticating PDF documents. I’ve been interested in getting a digital signature for business documents and noticed it’s often handled by the same certificate authorities that provide SSL certificates. What are the current trends in document certification, and why do we use the same providers for PDFs as we do for websites? Also, what does a digital signature actually prove to the recipient, and how does it compare to other authentication methods like passkeys or physical USB keys?

TRANSCRIPT

Corn

Hey everyone, welcome back to My Weird Prompts. I am Corn, and I am joined as always by my brother, Herman Poppleberry.

Herman

Hello, hello! It is great to be here. We have a really meaty one today, Corn. Our housemate Daniel was actually wrestling with this just the other day. He was sitting at his desk, surrounded by three different laptops and a pile of USB dongles, looking like he was trying to hack into a nineteen nineties mainframe.

Corn

I saw that. He was trying to open a government document on his Linux machine, right? And if you have ever tried to deal with official P-D-Fs on Linux, you know the struggle is real. It is like the final frontier of desktop Linux. Everything works perfectly—your drivers are fine, your games run through Proton, your development environment is a dream—until you need to sign a tax form or verify a government decree. Suddenly, you are hunting for proprietary libraries just to see a signature.

Herman

Exactly. And Daniel's prompt really gets to the heart of that frustration. He noticed that the signature on this Israeli government P-D-F was backed by a Certificate Authority, the same kind of people who sell S-S-L certificates for websites. He wanted to know why we use the same providers, what that signature actually proves, and how it compares to things like passkeys or those physical U-S-B keys that seem to be multiplying in everyone's desk drawers.

Corn

It is a great question because we often treat a digital signature as just a fancy version of an image of our handwriting, but underneath the hood, it is pure math and a very specific kind of trust. So, let's start with the basics, Herman. When Daniel sees that blue bar at the top of a P-D-F saying the document is certified, what is actually happening? What is the technology doing?

Herman

Right, so we need to separate the visual representation from the cryptographic reality. The little squiggly line that looks like your name? That is just a picture. It has zero legal or technical weight on its own. You could put a picture of a potato there and the cryptography would still work. The actual digital signature is a piece of metadata embedded in the P-D-F file structure. It relies on something called Public Key Infrastructure, or P-K-I.

Corn

P-K-I is one of those terms that sounds intimidating, like something you would hear in a high-level security briefing, but it is basically just a pair of keys, right?

Herman

Precisely. It is the foundation of almost everything secure on the internet. You have a private key, which you keep secret and never share with anyone, and a public key, which you give to everyone. They are mathematically linked in a way that what one key locks, only the other can unlock. When you sign a document, the software takes a hash of the entire file. A hash is like a unique digital fingerprint. If you change even one comma or add a single space to that document, the hash changes completely.

Corn

Okay, so the hash proves the document hasn't been tampered with. It is like a wax seal on an envelope. If the seal is broken, you know someone looked inside. But how does it prove I was the one who signed it?

Herman

That is where the private key comes in. Your signing software takes that hash—that digital fingerprint—and encrypts it using your private key. The result of that encryption is the digital signature. Now, anyone who has your public key can decrypt that signature and get the original hash back. Then, their software calculates the hash of the document they are looking at right now. If the two hashes match, it proves two things: first, that the document hasn't been altered since it was signed, and second, that only someone with your specific private key could have created that signature. It is called non-repudiation.

Corn

Non-repudiation. That is a big word. It basically means you can't say, "It wasn't me," because the math says it had to be you.

Herman

Exactly. Unless, of course, someone stole your key, which is why the hardware part of Daniel's question is so important. But before we get to the U-S-B sticks, let's address his point about the Certificate Authorities. Why is it that when I go to buy a document signing certificate, I'm looking at names like DigiCert or Sectigo or GlobalSign? Why is the same company that secures my browser connection also the one telling the world that my P-D-F is legitimate?

Corn

It feels like a monopoly on trust, doesn't it?

Herman

It is more like an economy of scale for trust. Think about it this way: if I give you a public key and say, "Hey Corn, this is my key," you might believe me because we have been brothers for decades. But if someone from across the world sends you a signed document, how do you know the public key actually belongs to them? You need a third party that both of you trust to vouch for that identity. These Certificate Authorities, or C-As, are that digital notary.

Corn

And they have to be incredibly reliable for this to work.

Herman

Oh, absolutely. These companies go through what are called WebTrust audits. They are incredibly rigorous, multi-month examinations of their physical security, their software, their hiring practices, everything. Because they have passed these audits, their own "root certificates" are pre-installed in your operating system and your browser. When you buy a certificate from them, they are performing a "vetting" process. For a basic individual certificate, they might just check your email. For a high-assurance business certificate, they are checking government I-Ds, business registrations, and sometimes even doing a video call to prove you are a real human.

Corn

So because Adobe and Microsoft and Apple already trust DigiCert's root certificate for web traffic, it makes sense to use that same established infrastructure for documents. If they trust them to verify that Google dot com is actually Google, they can trust them to verify that Herman Poppleberry is actually Herman Poppleberry.

Herman

Right. But there is a subtle difference in the standards. For a website, you use T-L-S certificates. For documents, we use the X-five-oh-nine standard, but with specific extensions for document signing. And this brings us to the "Blue Bar" Daniel saw. Adobe has something called the A-A-T-L, or the Adobe Approved Trust List. It is a private club of C-As that Adobe has personally vetted to meet their specific requirements for P-D-Fs. If your C-A isn't on that list, Adobe Acrobat will show a big yellow question mark or a red X, even if the signature is mathematically perfect.

Corn

That explains why Daniel was so frustrated. He was probably using a perfectly valid government certificate, but because he was on Linux using an open-source viewer, it didn't have access to Adobe's private list of trusted partners.

Herman

Precisely. On Linux, viewers like Okular or Evince rely on a different set of trust anchors. They often use the N-S-S database—that is Network Security Services—which is the same one Firefox uses. So if you want your Linux P-D-F viewer to recognize a signature, you often have to manually import the root certificate into your Firefox certificate store first. It is a very "Linux" solution to a very corporate problem.

Corn

It sounds like a nightmare for the average user. But Daniel also mentioned the physical U-S-B keys. In Israel, and in many other places like Estonia or even for certain U-S government contractors, you can't just have a file on your computer. You get a physical token. Why the extra step?

Herman

This is where we get into the different tiers of electronic signatures. Under the European e-I-D-A-S regulation, which is the gold standard for this stuff, there are three levels. First, you have the Simple Electronic Signature. That is just typing your name at the bottom of an email. It has very little legal weight. Then you have the Advanced Electronic Signature, or A-E-S. This uses P-K-I and a certificate, but the key might just be a file on your laptop.

Corn

And the third one is the big one, right?

Herman

Yes, the Qualified Electronic Signature, or Q-E-S. To get a Q-E-S, your private key must be stored on what is called a Qualified Signature Creation Device. That is usually a physical U-S-B token or a smart card that meets the FIPS one-forty-two or one-forty-three security standards. These devices are actually tiny computers. They are Hardware Security Modules, or H-S-Ms. The private key is generated inside the chip on that U-S-B stick, and it is designed so that the key can never, ever leave the chip. You can't copy it. You can't back it up. If you lose the stick, the key is gone forever.

Corn

So when Daniel clicks "sign," the P-D-F doesn't actually see his key?

Herman

Never. The computer calculates the hash of the document and sends that tiny string of data to the U-S-B stick. The stick asks Daniel for a P-I-N, signs the hash internally using the private key, and sends only the finished signature back to the computer. This provides the highest level of security because even if Daniel's computer is infected with the most advanced malware in the world, the hacker can't steal the key. They might be able to trick him into signing one wrong document while the stick is plugged in, but they can't take the identity with them.

Corn

I can see why governments love that, but the user experience is just brutal. Daniel was complaining about the "jumble" of keys. He has a YubiKey for his logins, a government key for his taxes, maybe another key for work. It feels like we are going backward to a world where we carry around a physical ring of digital keys. Is there any move toward making this easier in twenty twenty-six?

Herman

There is, and it is actually the biggest trend in the industry right now. It is called Remote Signing or Cloud H-S-M. Instead of you carrying the U-S-B stick, the Certificate Authority keeps your private key in a massive, high-security hardware module in their data center. When you want to sign something, you authenticate to them using your phone—usually with a biometric scan like Face I-D or a fingerprint. They then use your key to sign the document on their end and send it back.

Corn

Wait, doesn't that mean I have to trust the C-A with my private key? Doesn't that break the whole "only I have the key" rule?

Herman

It is a trade-off. But under e-I-D-A-S two point zero, which is being rolled out across Europe right now, these Remote Signing Service Providers are heavily regulated. They have to prove that they cannot use your key without your explicit, multi-factor authorization. This is the backbone of the new European Digital Identity Wallet. The goal is a widespread rollout of the European Digital Identity Wallet across the EU in the coming years, allowing citizens to use their phone as a Qualified signature device. No more U-S-B sticks, no more smart card readers. Just your phone and your face.

Corn

That sounds a lot like Passkeys. Daniel asked about the comparison there. If I use a Passkey to log into my bank, is that the same thing as digitally signing a document?

Herman

They are cousins, but they have different jobs. Passkeys are based on the F-I-D-O-two standard. They are amazing for authentication—proving to a server right now that you are you so you can log in. They are fast and they stop phishing. But a P-D-F signature is about long-term validation. A Passkey login is a conversation between you and a website that happens in milliseconds. A signed P-D-F is an object that might sit in a digital archive for fifty years.

Corn

So the P-D-F signature has to carry its own proof with it.

Herman

Exactly. When you sign a P-D-F at the highest level—what we call P-Ad-E-S, or P-D-F Advanced Electronic Signatures—the file includes something called Long-Term Validation information, or L-T-V. This embeds the entire chain of trust right into the file. It includes the C-A's certificate, the proof that your certificate wasn't revoked at the exact second you signed it, and a trusted timestamp from a third party. This means that twenty years from now, even if the C-A has gone out of business and the internet has changed completely, someone can open that file offline and mathematically prove it was valid when it was signed.

Corn

That is a huge distinction. A Passkey gets you through the door, but a P-D-F signature is the permanent record of what you did once you were inside. Now, I want to go back to Daniel's Linux problem for a second. Why is it that even in twenty twenty-six, we are still struggling with this? If it is all just math, why does the software matter so much?

Herman

It is the "middleware" problem. To talk to those U-S-B keys or smart cards, your computer needs a driver, but it also needs a library called P-K-C-S eleven. On Windows, this is all handled by the Microsoft Crypto-A-P-I, and it is mostly invisible. On Linux, you have to make sure the `pcscd` service is running, you need the `opensc` package, and then you have to tell your P-D-F viewer exactly where the S-O file for the driver is located. If any of those pieces are missing or the versions don't match, the whole thing falls apart. It is a classic case of the technology being ready but the desktop integration being fragmented.

Corn

It’s also the Adobe factor, right? They basically wrote the book on P-D-Fs.

Herman

They did. For a long time, the P-D-F spec was proprietary. Now it is an I-S-O standard, but Adobe still has a huge influence. They created the A-A-T-L and the E-U-T-L—the European Union Trust List—integration in Acrobat. If you are a developer making an open-source viewer, you are constantly playing catch-up with how Adobe implements new features. For example, Adobe recently started pushing for "Seals" instead of "Signatures."

Corn

What is the difference between a seal and a signature?

Herman

A signature is for a person—Herman Poppleberry. A seal is for a legal entity—The My Weird Prompts Podcast Corporation. Seals are used for automated things, like when a bank issues a million monthly statements. They don't want a person to sign each one, so they use an electronic seal. It provides the same integrity and authenticity, but it represents the organization rather than an individual.

Corn

That makes sense. Now, Herman, we have to talk about the elephant in the room. We are in early twenty twenty-six, and everyone is talking about quantum computers. Daniel’s prompt didn't mention it, but I know you’ve been losing sleep over it. If our current signatures are based on R-S-A or Elliptic Curve math, and quantum computers can break that math, are all our signed documents about to become worthless?

Herman

It is a massive concern. There is a strategy called "Harvest Now, Decrypt Later." Hackers are stealing signed P-D-Fs today and just storing them, waiting for a quantum computer powerful enough to crack them in five or ten years. Once they can do that, they can forge signatures on old contracts or read encrypted government files. This is why the industry is moving toward P-Q-C, or Post-Quantum Cryptography.

Corn

I remember you mentioning Dilithium earlier. Is that still the plan?

Herman

Yes! N-I-S-T, the National Institute of Standards and Technology, finalized the standards in August twenty twenty-four. They gave them new names. Dilithium is now officially M-L-D-S-A, which stands for Module-Lattice-Based Digital Signature Algorithm. Kyber is now M-L-K-E-M for key encapsulation. In twenty twenty-six, we are seeing the first "Hybrid" signatures. A C-A will sign your P-D-F twice: once with traditional R-S-A so today's software can read it, and once with M-L-D-S-A so that it remains secure even when quantum computers arrive.

Corn

That is incredible. We are literally building a bridge to a future where the current laws of digital physics might not apply. It really highlights why you need these big Certificate Authorities. You or I can't implement a hybrid M-L-D-S-A signature on our own. We need an organization whose entire job is to stay ahead of that curve.

Herman

Exactly. You are paying for the research and the infrastructure. You are paying for someone to watch the N-I-S-T announcements so you don't have to. And you are paying for the revocation system. If a C-A realizes one of their root keys has been compromised, they have to be able to tell every computer in the world to stop trusting it within minutes. That ecosystem of revocation—using things like O-C-S-P, the Online Certificate Status Protocol—is what makes the whole thing work.

Corn

So, to wrap this up for Daniel and everyone else listening: a digital signature proves the document hasn't changed, it proves who you are because a trusted third party vetted you, and it makes the signature legally binding in a way that is very hard to take back. It is more secure than a Passkey for long-term records, but it is currently more of a headache to use, especially if you are a Linux fan.

Herman

But it is getting better. Between the E-U Digital Identity Wallet and the move toward cloud-based signing, the era of the "U-S-B jumble" is coming to an end. We are moving toward a world of "Verifiable Credentials." Instead of sending a P-D-F of your university diploma, you will send a tiny digital token that the employer's system can verify instantly against the university's public key. The document itself becomes secondary to the data it contains.

Corn

I love that. A world with no more "sub-par" P-D-F support on Linux because the P-D-F itself isn't the point anymore. We can dream, Herman. We can dream.

Herman

Until then, keep your `pcscd` running and your Firefox certificate store clean.

Corn

Words to live by. Well, I think we’ve covered the "what" and the "why" of it. Thanks to Daniel for the prompt that sent us down this rabbit hole. It is questions like these that keep us curious about the invisible plumbing of the internet.

Herman

It really is. We’ve been doing this for three hundred twenty-five episodes now, and the community just keeps growing. If you have your own weird prompt, or if you’ve spent your afternoon trying to get a smart card reader to work on Arch Linux, we want to hear from you.

Corn

You can find us at our website, myweirdprompts dot com. We have a contact form there, and you can browse our full archive of episodes. And if you are listening on Spotify or Apple Podcasts, please leave us a review. It genuinely helps the show reach more people who are interested in the technical nuances of our world.

Herman

It really does. We appreciate every single one of you.

Corn

Thanks for joining us, Herman. This has been My Weird Prompts. I'm Corn.

Herman

And I'm Herman Poppleberry. We will see you next time!

Corn

Stay curious, everyone. Bye!