MY WEIRD PROMPTS

Podcast Transcript

EPISODE #368

The Geometry of Secrets: How SSH Keys Protect the Web

Published January 30, 2026 • Runtime: 21:02

https://myweirdprompts.com/episode/math-of-ssh-key-security/

EPISODE SYNOPSIS

In this episode of My Weird Prompts, Herman and Corn explore the fascinating mathematics behind SSH keys, moving from the prime factorization of RSA to the sophisticated geometry of Elliptic Curve Cryptography (ED25519). They explain why deriving a public key from a private one is a simple calculation while the reverse would take longer than the life of the universe, illustrating the "trapdoor functions" that secure our global infrastructure. From the mechanics of digital handshakes to the physical risks of power analysis attacks, this deep dive reveals how the invisible world of number theory keeps your data safe from even the most powerful supercomputers.

DANIEL'S PROMPT

Daniel

Hi Herman and Corinne. We've talked in previous episodes about authentication and what will come after two-factor authentication. The mainstay of technical authentication is the SSH key. I find it intriguing that you can infer a public key from a private key, but not vice versa. I'd love to understand exactly how that works. We've discussed checksums before, which this reminds me of. I'd love to learn more about how SSH keys provide authentication for technical services and act as the backbone of security in technology.

TRANSCRIPT

Corn

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

Herman

Herman Poppleberry, at your service. And Corn, I have to say, I am feeling particularly energized today. Our housemate Daniel sent us a prompt that is right up my alley. It is one of those things that most people use every single day without ever thinking about the beautiful mathematics happening under the hood.

Corn

Yeah, Daniel was actually looking at some keys he generated recently and it sparked this whole train of thought about how we prove who we are to machines. He was specifically looking at an elliptic curve key, which I know you have been wanting to talk about for months.

Herman

Guilty as charged. It is fascinating because most technical users know they need an S S H key to talk to GitHub or a remote server, but the mechanism of why a public key can be derived from a private key, while the reverse is practically impossible, feels like magic. It is the foundation of almost everything we do online.

Corn

It really is. And Daniel mentioned something interesting. He said it reminded him of our discussion on checksums from a few episodes back. Remember when we talked about how a tiny change in a file completely alters the hash? He is sensing a connection there, and I want to dig into whether that is a fair comparison or if we are looking at a different beast entirely.

Herman

It is a brilliant intuition, actually. There is a shared D N A of one way mathematical functions. But before we get into the weeds of the math, maybe we should set the stage for why this matters. If you are a developer or a system administrator, S S H keys are your digital passport. They are the backbone of secure communication.

Corn

Right, and unlike a password, which you send over the wire and hope the other side handles it correctly, an S S H key pair stays with you. Well, the private part does. So, Herman, let us start with the basics. What is actually happening when Daniel runs a command to generate a new key?

Herman

When you generate a key pair, you are essentially asking a computer to pick a very large, very random number. That is your private key. Then, the computer performs a specific mathematical operation on that number to produce a second number, which is your public key. The crucial part, the part that makes the whole internet work, is that this operation is a trapdoor function.

Corn

A trapdoor function. I love that term. It sounds like something out of an adventure movie.

Herman

It sort of is! It is a function that is easy to compute in one direction but incredibly difficult to reverse unless you have a special piece of information. In the case of older R S A keys, that math was built on the difficulty of factoring the product of two massive prime numbers. If I give you two primes, it is easy to multiply them. If I give you a number that is five hundred digits long and tell you it is the product of two primes, you could spend the rest of your life trying to find them.

Corn

So that is the one way street. But Daniel specifically mentioned the E D two five five one nine algorithm. That is not R S A, right? That is the newer, shinier version we see everywhere now.

Herman

Exactly. That is elliptic curve cryptography. Instead of just multiplying big primes, we are using the geometry of a specific curve. Imagine a complex graph with a curve on it. We pick a starting point, called the base point, and then we perform a series of jumps or reflections across that curve based on our private key. The final point we land on is the public key.

Corn

Okay, so the public key is the destination, and the private key is the set of instructions for how many jumps to take?

Herman

Precisely. In math terms, we call this scalar multiplication. It is easy to do the jumps to find the destination. But if I just show you the starting point and the ending point, finding out how many jumps I took is what we call the Elliptic Curve Discrete Logarithm Problem. And because of the way the curve is shaped, there is no known shortcut. You would have to check every single possibility. And when we say every possibility, we are talking about numbers larger than the number of atoms in the observable universe.

Corn

That is the scale that always breaks my brain. We talk about security being hard to crack, but it is not just hard like a difficult puzzle. It is hard like trying to find one specific grain of sand on a planet made entirely of sand.

Herman

It is even more lopsided than that. If you tried to brute force an E D two five five one nine key with all the computing power currently on earth, the sun would likely burn out before you made a dent. That is why it is the backbone. We trust the math because the universe literally does not have enough energy or time to reverse it.

Corn

So, let us go back to Daniel's point about checksums. In episode three hundred fifty eight, we talked about how triage and chaos theory relate to medical data, and we touched on how checksums verify integrity. A checksum is also a one way function. You take a file, run it through an algorithm, and get a string of characters. You cannot turn those characters back into the file. How is that different from what a public key does?

Herman

That is such a sharp observation. The similarity is that they are both one way. But the purpose and the structure are different. A checksum, or a hash function, is designed to be a fingerprint. It is meant to represent a large amount of data in a small, fixed size string. If you change one bit of the original data, the fingerprint changes completely. But a hash function does not have a pair. There is no public hash and private hash that work together to prove identity.

Corn

Right, a checksum just says this thing has not changed. It does not necessarily prove who sent it or that the sender has a secret.

Herman

Exactly. S S H keys use asymmetric encryption. That word asymmetric is the key. It means the key used to lock the door is not the same as the key used to open it. With a checksum, there is no locking or unlocking. It is just a label. With an S S H key, I can give my public key to every server in the world. I can post it on my website. It does not matter. Because the only thing that public key can do is verify that something was signed by my private key.

Corn

So, let us walk through that handshake. If I want to log into a server using my S S H key, I am not actually sending my private key to the server, am I? That seems like it would be a huge security risk.

Herman

Oh, absolutely not. If you ever send your private key over a network, you have already lost. The handshake is much more clever. When you try to connect, the server says, hey, I have your public key on file. I am going to send you a random challenge. It is basically a piece of digital gibberish.

Corn

And I have to prove I can handle that gibberish?

Herman

Exactly. Your computer takes that challenge and signs it using your private key. This creates a digital signature. You send that signature back to the server. The server then uses your public key to verify the signature. If the math checks out, the server knows for a fact that you possess the private key without you ever having to show it. It is like a secret knock where the knock changes every single time you show up at the door.

Corn

This really explains why it is the backbone of technical services. Think about how many automated systems are running right now. Servers talking to other servers, deployment scripts pushing code to the cloud. You cannot have a human there typing in a two factor authentication code every five seconds.

Herman

You really can't. And that is actually a misconception people have. They think S S H keys are less secure than two factor authentication because there is no second step. But in reality, an S S H key is a form of something you have. It is much stronger than a password, which is something you know. If your private key is protected by a strong passphrase, it is essentially two factor authentication in one package. You have the physical file on your machine, and you know the passphrase to unlock it.

Corn

But what about the risk of that file being stolen? If someone gets onto my laptop and copies my private key, they are me. They have my digital identity for every service I have ever connected to.

Herman

That is the nightmare scenario, and it is why key management is so important. It is also why the industry is moving toward things like hardware security modules or YubiKeys. These are physical devices that generate and store the private key in a way that it can never be exported. When you need to sign something, the data goes into the device, the device signs it, and the signature comes out. The private key never even touches your computer's memory.

Corn

That is fascinating. So the math remains the same, but we are just getting better at building physical cages for the private key so it cannot escape.

Herman

Precisely. And Daniel's mention of the E D two five five one nine algorithm is relevant here too. One of the reasons it has become so popular, besides being fast, is that it is designed to be resistant to certain types of side channel attacks. Older algorithms like R S A can sometimes leak tiny bits of information about the private key based on how much power the processor uses or how long the calculation takes.

Corn

Wait, really? You can steal a key by measuring how much electricity a computer is using?

Herman

In a lab setting, yes! It is called a power analysis attack. If the math takes slightly longer to process a one than a zero in binary, a sophisticated attacker can reconstruct the key. But the newer elliptic curve algorithms are designed to be constant time. No matter what the key is, the math takes exactly the same amount of time and energy. It closes that physical loophole.

Corn

That is the kind of detail that makes me realize how deep this rabbit hole goes. It is not just about the abstract math being right. It is about how that math translates into the physical world of electrons and silicon.

Herman

It really is a multidisciplinary field. You have number theory, computer science, and electrical engineering all shaking hands to make sure your GitHub push is secure.

Corn

So, Herman, I want to go back to the idea of inferring the public key from the private key. Daniel mentioned he was intrigued that it only works one way. In his prompt, he noticed that when he puts his private key into a manager, it just knows the public key. Why is that? Is the public key actually hidden inside the private key file?

Herman

That is a common point of confusion. In many formats, like the Open S S H format, the public key is actually bundled inside the private key file for convenience. But even if it weren't, the math allows you to derive it instantly. Remember our analogy of the jumps on the curve? If the private key is the number of jumps, and you know the starting point which is a standard part of the algorithm, you just perform the jumps and you get the public key.

Corn

Okay, so deriving the public key is just doing the math as intended. But going the other way is like trying to un-bake a cake to find the original ingredients.

Herman

Exactly. Or like trying to find the two prime numbers that made a massive product. There is no shortcut. You just have to guess and check, and the number of guesses required is so large that it is effectively impossible. This is what we call computational irreducibility in some contexts, though that is a bit of a stretch. In cryptography, we just call it a hard problem.

Corn

I love the idea that our entire global economy and security infrastructure are resting on the fact that some math problems are just really, really hard to solve. It feels almost fragile when you put it that way.

Herman

It does feel fragile! And that is why cryptographers are always looking over their shoulder at quantum computing. If we ever build a sufficiently powerful quantum computer, it could use something called Shor's algorithm to factor those large primes or solve the elliptic curve problem in minutes instead of billions of years.

Corn

I was going to ask about that. Is that why we are seeing people talk about post-quantum cryptography?

Herman

Exactly. We are already looking for new hard problems. In fact, just a couple of years ago in late twenty twenty four, N I S T finalized the first official standards for post quantum cryptography. They have names like M L K E M and M L D S A. These are built on lattice based math, which even a quantum computer would find difficult. We are already seeing hybrid systems where an S S H connection uses both a classical key and a quantum resistant one just to be safe.

Corn

But for now, here in early twenty twenty six, the S S H key is still the king. I think one of the things that makes it so resilient is how simple the interface is for the user. Once you set it up, it just works. But the implications of how we manage those keys are huge.

Herman

They really are. One of the biggest mistakes I see people make is using the same S S H key for everything. Their personal GitHub, their work servers, their private cloud. If that one key is compromised, every door is open.

Corn

So the takeaway there is compartmentalization?

Herman

Absolutely. You should have different keys for different contexts. It is also good practice to rotate them. Just because a key is mathematically secure doesn't mean it hasn't been leaked through a backup or a lost device.

Corn

I also think there is a broader lesson here about how we think about digital identity. We are moving away from the era of the password. Whether it is S S H keys for developers or Passkeys for the general public, we are moving toward a world where identity is proven through possession of a cryptographic secret rather than knowledge of a string of text.

Herman

And that is a much more secure world. Passkeys, which many of our listeners might be using on their phones or laptops now, are essentially built on the same principles as the S S H key Daniel was looking at. They use asymmetric cryptography to prove you are you without ever sending a secret over the internet.

Corn

It is funny how the most advanced tech often circles back to these fundamental mathematical truths. Daniel's curiosity about why the math only works one way is really the central question of modern security. If it worked both ways, we wouldn't have an internet. Not a secure one, anyway.

Herman

We really wouldn't. Every bank transaction, every private message, every software update relies on that one way street. It is the invisible architecture of our lives.

Corn

I am curious about the practical side of this for someone who isn't a developer. If you are a regular listener and you hear us talking about S S H keys and elliptic curves, how does this affect you? I think the answer is that it gives you a framework for why things like two factor authentication and hardware keys are so important. It is not just an annoying extra step; it is a fundamental shift in how security works.

Herman

Right. And it also helps you understand why you should never, ever share certain things. If a service asks for your private key, they are either incompetent or malicious. There is never a legitimate reason for a third party to have your private key.

Corn

That is a great rule of thumb. The public key is for the world; the private key is for you and you alone.

Herman

Exactly. It is like your signature versus your hand. You can give your signature to anyone to prove you agree to a contract, but you don't give them your hand.

Corn

That is a bit of a gruesome analogy, Herman, but it works!

Herman

I try my best. But seriously, the fact that Daniel noticed this while just working in his terminal is what I love about this show. These profound truths are hiding in plain sight in our everyday tools.

Corn

They really are. And I think it is important to acknowledge that while the math is robust, the human element is always the weakest link. We can have the most perfect elliptic curve in the world, but if someone leaves their private key in a public cloud storage bucket, the math doesn't matter.

Herman

That is the tragedy of security. You are only as strong as your worst habit. But having these keys as the backbone means that when we do our part, the technology actually has our back. It is a partnership between human discipline and mathematical certainty.

Corn

I think that is a perfect place to start wrapping up this part of the discussion. We have covered the math, the difference between keys and checksums, the handshake, and the future risks. What are the big takeaways you want people to have?

Herman

First, realize that your digital life is built on these one way mathematical streets. Second, treat your private keys like the crown jewels because, in a digital sense, they are. And third, don't be afraid of the technical details. Understanding even a little bit of how an S S H key works makes you a much more informed and secure citizen of the internet.

Corn

And if you are a developer like Daniel, maybe take a second to appreciate that E D two five five one nine key next time you generate one. It is a tiny, elegant piece of math doing a massive amount of work.

Herman

It really is. It is the silent guardian of the terminal.

Corn

Well, this has been a deep dive. I feel like I understand my own laptop a little better now. And to our listeners, if you have been enjoying these deep dives into the weird and wonderful world of tech and science, we would really love it if you could leave us a review on your podcast app or on Spotify. It genuinely helps other curious minds find the show.

Herman

It really does. We see every review, and it fuels our curiosity to keep digging into these prompts. So thank you to everyone who has already left one.

Corn

And thank you to Daniel for sending this one in. It is always fun to explore the things we usually take for granted. You can find all our past episodes, including the ones we mentioned today, at myweirdprompts.com. We have got a full R S S feed there and a contact form if you want to send us your own weird prompts.

Herman

We are also on Spotify, so you can follow us there to make sure you never miss an episode. We have got some really interesting ones coming up in the next few weeks.

Corn

Definitely. We are going to be looking at some biology prompts soon that I think are going to blow some minds. But for today, that is our show.

Herman

Until next time, keep asking those weird questions.

Corn

This has been My Weird Prompts. Thanks for listening.

Herman

Goodbye everyone!