



## HOTP & TOTP

**Description:** Meta winds down third-party content filtering. Is encryption soon to follow? Taking over abandoned command-and-control server domains (strictly for research purposes only!). IoT devices to get the "Cyber Trust Mark." Will anyone notice or care? Synching receives a (blessedly infrequent) update. Government email is not using encryption? Really? Email relaying prevents point-to-point end-to-end encryption and authentication. Just because Let's Encrypt doesn't support email doesn't mean it's impossible. What sci-fi does ChatGPT think I (Steve) should start reading next? To auto-update or not to auto-update? Is that one question or two? Until today, we've never taken a deep dive into the technology of time-varying six-digit one-time tokens. Let's fix that! Also, last week's uncaptioned picture is finally captioned!

High quality (64 kbps) mp3 audio file URL: <http://media.GRC.com/sn/SN-1008.mp3>

Quarter size (16 kbps) mp3 audio file URL: <http://media.GRC.com/sn/sn-1008-lq.mp3>

---

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We have one of Steve's favorite propeller-head episodes. There's of course some security news, the answer to some of your questions. And then he's going to go into - he's actually going to answer a listener question from Max who says, "Yeah, I look at my 2FA, my authenticator app, and it always seems to be, like, nonrandom repetitions of numbers in there. Are these numbers really random?" It's an interesting question, but it prompted Steve to go into a deep dive on how these one-time time-based passwords are generated. And you won't believe what a kludge it is. It's mind-bending. Stay tuned. Security Now! is coming up next.

**Leo Laporte:** This is Security Now! with Steve Gibson, Episode 1008, recorded Tuesday, January 14th, 2025: HOTP and TOTP.

It's time for Security Now!, the show where we cover your security, your privacy online. We cover all of the stuff that you want to know about with computing because this guy right here is the king of computing, Mr. Steve Gibson, GRC.com. Hi, Steve.

**Steve Gibson:** Hello, my friend.

**Leo:** I was getting the fire report from you, and you're safe and sound.

**Steve:** Yes. I'm safe. A number of our listeners wrote to say, hey, you know, all I know about you is that you're somewhere in California. Are your feet toasty? And the good

news is we're about an hour and a half south of the, well, I don't want to call it the "action" or "excitement" or anything. I mean, it's devastation.

**Leo:** It's so horrible, yeah.

**Steve:** And Leo, the cost. That's what's astonishing to me. You know, I mean, we've noted that anything that is done these days is expensive. So, I mean, the cost throughout the entire system of rebuilding this much area is just, you know, astonishing.

**Leo:** Well, and there's of course the question of whether it even should be rebuilt, given the hazards.

**Steve:** Right. Why do people keep building in Florida, where hurricanes just keep coming through and wiping the real estate clean?

**Leo:** Because we're human. We don't give up easy.

**Steve:** It's nuts.

**Leo:** So what's coming up today on Security Now!?

**Steve:** Okay. So I was tempted after I learned the word, where is it, apophenia, to give the podcast that title, but even I can't remember that word. So I thought, no, that's not good. Instead, I'm titling today's second podcast of the year, 1008 for January 14th is titled HOTP & TOTP.

**Leo:** Well, that's much clearer than apophenia.

**Steve:** Yeah.

**Leo:** But why apophenia? What's apophenia when it's at home? Did we talk about that last week?

**Steve:** No, it's the tendency to see patterns in random noise where none exist.

**Leo:** Ah. That's a good term.

**Steve:** And so today's topic derives from actually two pieces of email that I received from a listener two weeks apart, suspicious of the digits that his authenticator was generating. And I realized that through all 1007 previous episodes, although we've talked about time-based one-time passwords, and in the case of HOTP HMAC-based one-time passwords, we've never looked at the actual algorithms that are used. And so what we have today under the simple-seeming acronyms HOTP and TOTP, well, it's going to give

our listeners a workout because it turns out I had a lot to say. In fact, the show notes, as I mentioned to you before, are now at version 1.2. This is the second time I've had to put a version number on the show notes because I just, even though I sent them off to everybody in version 1.0, I guess the early evening on Monday, I just couldn't stop messing with them.

So then I went back to work, and I fixed some more and fleshed some more things out. And then at the end of the evening I sent - I updated, I didn't send another email, but I updated the online versions to 1.1. And then as I was laying in bed last night thinking, you know, I really didn't explain why...

**Leo:** Oh, crud.

**Steve:** ...exactly why you wouldn't have an even distribution of outcomes, I just said really pretty good, but I didn't explain why it wouldn't be perfect.

**Leo:** Not perfect. Not perfect.

**Steve:** No. So there went this morning, up until, well, yeah, pretty much all morning, and whereupon 1.2 was updated on the website.

**Leo:** I do understand the apophenia because I often feel like, oh, that number is not random. You know, that's too obvious; right? But that's the nature of randomness.

**Steve:** And I'll tell you, Leo, when I see times on the clock like 2:56 and 5:12 and 10:24, I think, wait a minute.

**Leo:** Hey.

**Steve:** That's a power of two. That's one of my special numbers.

**Leo:** You're funny.

**Steve:** And frequently will look at the clock, and it'll be 11:11.

**Leo:** 11:11, I know.

**Steve:** What are the chances that...

**Leo:** That happens to you, too. Because I see 11:11 all the time.

**Steve:** Yeah, see, I think we're actually - there's many more of those than we suspect.

**Leo:** Oh, maybe that's it.

**Steve:** That's the only explanation; right? It couldn't be apophenia because...

**Leo:** Could be apophenia.

**Steve:** Yeah. Anyway...

**Leo:** I love it.

**Steve:** So but that's going to be, again, if anyone's mowing the lawn, I would say pull off to the side of the garden when we get to this.

**Leo:** Do not operate heavy machinery, is that what you're saying?

**Steve:** Yeah. And if you do have a self-driving car that you trust, well, then maybe it's safe to continue listening.

**Leo:** Okay.

**Steve:** But we're going to first talk about Meta briefly, winding down third-party content filtering as our segue into Matthew Green, the cryptographer's comment about that, whether encryption is soon to follow. Taking over abandoned command-and-control server domains (now, strictly for research purposes only!). We also have IoT devices getting the "Cyber Trust Mark" and wonder whether anyone will anyone notice or care? Syncting received a blessedly infrequent update, which we'll touch on. And government email is still not using encryption. Really? Also email relaying prevents point-to-point, end-to-end encryption and authentication that we're going to look at. And just because Let's Encrypt doesn't support email doesn't mean it's impossible. Also, we're going to examine what sci-fi ChatGPT thinks that I, Steve, should start reading next, and I have a new author as a consequence.

**Leo:** Oh, my gosh, it worked.

**Steve:** Also, to auto-update or not to auto-update? And is that one question or two? And until today, as I said, somehow through 1007 episodes before this, we've never actually looked at the technology that produces the - remember the football? That was like our first introduction to a time-varying code, those time-varying six-digit time tokens. And we're going to find out whether they're as random as they appear, or maybe more random than they don't appear? Anyway, I'm not sure. We do have a great Picture of the Week which it just sort - I didn't even caption it. This one is just so good, it doesn't even need a caption. And then we do have the result from last week's uncaptioned, come-up-with-a-caption picture contest. So I think certainly a gripping and interesting podcast for our listeners.

**Leo:** Goodness gracious, yes.

**Steve:** And boy, I'll tell you, you're going to be done by the time this podcast is over. You're going to be crispy.

**Leo:** All right. Well, I'm going to grab a cup of coffee.

**Steve:** You're going to be crispy.

**Leo:** So I get my thinking cap on. Steve Gibson is going to do it again, kids. Stay tuned. This is why we look forward to Tuesdays, because Tuesday is Steve Gibson GRC Day. All right. Picture of the Week time, Mr. G.

**Steve:** Which needs no caption.

**Leo:** Okay.

**Steve:** And it's just it speaks for itself.

**Leo:** All right. I've seen it now. Let me show everybody else. You want to describe this, Steve?

**Steve:** So we're looking at this so-called "end cap" that stands at the end of some sort of a store, like a hardware store or something that has aisles. And so this is at the end of one of the aisles. And it's got a big sign, and maybe it's seasonal because I see like some holly leaves and berries and some wispy, like some stars and some, maybe it's like signs of wind in the borders of this large sign. So over this end cap it says "Hard to Find Items," where "Items" is, you know, in 300-point font, really there. And we're looking at one, two, three, four, five, six shelves completely empty. So yes, indeed.

**Leo:** They're hard to find, all right. I can't see them anywhere.

**Steve:** They are difficult to find.

**Leo:** That's hysterical.

**Steve:** Yeah. Anyway, I just loved it because it just spoke for itself. And interestingly, one of our listeners asked one of the AIs, I don't know if it was Chat or whom, oh, there we go, or what.

**Leo:** Or what.

**Steve:** Yeah. But to describe the picture. And it produced a complete interpretation of even what would be found humorous about this picture.

**Leo:** Isn't that amazing.

**Steve:** And our listener's comment was, boy, you know, for unsighted people, this is going to be life-changing.

**Leo:** Huge, yeah.

**Steve:** Because, you know, it does the work of interpreting. It's not just like, oh, well, it is some shelves that are empty. It's like, okay.

**Leo:** I don't get it.

**Steve:** Anyway, it's fantastic.

**Leo:** Now, did ChatGPT come up with a caption for last week's picture?

**Steve:** No, it didn't. But we can now scroll to the next page, where you will find the caption actually I gave it. Now, recall that we have this lone gate just sort of sitting out in the middle of a meadow with beautiful forestation, shrubs and trees and things, behind it. Nothing else is there. Anyway, a listener of ours, Steven Kangas wrote to me, he said: "Caption Contest," and he suggested "Earth Abides."

**Leo:** Oh, yeah, what a wonderful book that was.

**Steve:** Right.

**Leo:** Did you ever read that book?

**Steve:** I never did.

**Leo:** Oh, I loved that. It's a classic.

**Steve:** He said: "From a great book about life of a small number of survivors after a devastating worldwide pandemic." And anyway, so that kind of put me on the sci-fi thread. And anyway, so I gave our picture of just this lonely gate standing there with nothing around it except nature, I said: "Some believe that, long ago, humans roamed this beautiful planet."

**Leo:** Awww. Great title.

**Steve:** And that's all that's left of us, and it might just as well be for the best.

**Leo:** Yeah. Got a gate. That's it. Yup.

**Steve:** At least from the planet's standpoint. Okay. So it wasn't until I encountered Matthew Green's comment about Meta's announcement last week that I decided to just touch on that, like on any of that today. So before we get to what Matthew posted, here's a brief update for those who may have been without any other source of news for the past week.

Last Tuesday, a week ago, while we were recording this podcast, or Podcast 1007, Mark Zuckerberg posted a video in coordination with a Meta news release titled "More Speech and Fewer Mistakes." Part of what they wrote under the heading "Ending Third-Party Fact-Checking Program, Moving to Community Notes," was, they wrote: "When we launched our independent fact-checking program in 2016, we were very clear that we didn't want to be the arbiters of truth. We made what we thought was the best and most reasonable choice at the time, which was to hand that responsibility over to independent fact checking organizations. The intention of the program was to have these independent experts give people more information about the things they see online, particularly viral hoaxes, so they were able to judge for themselves what they saw and read.

"That's not the way things played out, especially in the United States. Experts, like everyone else, have their own biases and perspectives. This showed up in the choices some made about what to fact-check and how. Over time we ended up with too much content being fact-checked that people would understand to be legitimate political speech and debate. Our system then attached real consequences in the form of intrusive labels and reduced distribution. A program intended to inform too often became a tool to censor.

"We're now changing this approach. We will end the current third-party fact-checking program in the United States and instead begin moving to a Community Notes program. We've seen this approach work" - and I'm so tempted to put "work" in quotes, but they didn't - "approach work on X, where they empower their community to decide when posts are potentially misleading and need more context, and people across a diverse range of perspectives decide what sort of context is helpful for other users to see. We think this could be a better way of achieving our original intention of providing people with information about what they're seeing, and one that's less prone to bias."

And then a bit lower down in the lengthy posting, Meta noted, they said: "As part of these changes, we will be moving the trust and safety teams that write our content policies and review content out of California to Texas and other U.S. locations." So that was the preamble which led Matthew Green, the well-known cryptographer at Johns Hopkins University, to I guess worry about what's happening. And he then sent, he said: "Lots of folks are commenting on the fact that Meta is cozying up to the current administration, cutting out fact-checking and other forms of moderation. This stuff is obviously worrying, but my big concern," he wrote, "is what happens when the government asks them to turn off encryption."

Okay now, what's interesting is that our own CISA, you know, the Cybersecurity Infrastructure Security Agency just published a five-page PDF titled "Mobile Communications Best Practice Guidance." And its first number one recommendation, like in the officially published five-page CISA Guide, is titled "Use Only End-to-End Encrypted Communications." Underneath they wrote: "Adopt a free messaging application for secure communications that guarantees end-to-end encryption, such as Signal or similar

apps. CISA recommends an end-to-end encrypted messaging app that is compatible with both iPhone and Android operating systems, allowing for text message interoperability across platforms.

"Such apps may also offer clients for macOS, Windows, and Linux, and sometimes the Web. These apps typically support one-on-one text chats, group chats with up to a thousand participants, and encrypted voice and video calls. Additionally, they may include features like disappearing messages and images which can enhance privacy. When selecting an end-to-end encrypted messaging app, evaluate the extent to which the app and associated services collect and store metadata."

And there was another related piece of news about this telecom hack which was sort of the underpinning for all this; right? The reporting is that a source has told The Wall Street Journal the names of three additional American telcos that were breached by the Chinese espionage group Salt Typhoon last year. Those are Charter, Consolidated Communications, and Windstream. As we know from previous reporting, the other four now known victims were AT&T, Lumen, Verizon, and T-Mobile, with those first three - ATT, Lumen, and Verizon - victoriously claiming a week or two ago that they are now, they have fully expunged the perpetrators from their networks. Now, given the count of breached firms that has been shared publicly, there are still two that remain unnamed. So two more telcos we don't yet know about.

So the clear point being made here is that no one can rely upon the security of telecommunications carriers to protect the privacy of anything that uses their bandwidth. So this begs the question, you know, whoever did believe that we could rely upon anyone else's security? After all, that's the entire point of, and reason for, communicating consumers adding our own after-the-fact pre-Internet encryption, as we called it long ago, to everything we care about, specifically so that we don't need to trust anyone else. And one of our early abbreviations, TNO (Trust No One), has been the rule of the road from the start.

So this brings us to Matthew Green's worries about encryption. And at this point that's all they are is worries, and I would suggest that it's probably not worth worrying about. No one appears to have any idea what the incoming Trump administration plans to do or will do. But I'm certain that Elon Musk, who appears to have a great deal of technical sway with president-elect Trump, certainly understands the pros and cons of any form of mandated backdoor being forced into today's exception-free end-to-end encryption, as we have it now. And I'm certain that our incoming president is well aware that the Chinese government appears to be behind much, if not most, of the hacking into our nation's critical infrastructure, and especially the government's infrastructure. Mr. Trump's feeling about China are well known, so I would be quite surprised if he wanted to, in any way, open any doors - or backdoors - into our nation's encrypted communications.

I would therefore be very surprised if anything were to change along the lines that Matthew fears. You know, changes in content moderation are not even in the same world as changes that would weaken our encrypted communications. And, you know, I think that much should be clear to everyone.

**Leo:** Good. I hope that's right, yeah.

**Steve:** I think he's just grumbling because he's worried about content moderation changing. But boy, you know, saying no to encryption, I don't think it'll happen because, again, I just think the downsides are too severe.

---



**Leo:** I mean, WhatsApp is using Signal's encryption. You feel like it's safe to use WhatsApp?

**Steve:** Yes. Yes. And again, we know that it's not actually necessary to crack encryption because the handsets that we're all using are receiving plaintext and emitting plaintext, receiving voice communications and video in the clear. And so what we're creating is a bulletproof, insofar as like to the best cryptographers in the industry know, a bulletproof tunnel. But at the input and output of the tunnel, everything is in the clear and is available to the underlying operating system.

**Leo:** Right. You don't have to go to all that trouble. Just get the phone.

**Steve:** Yeah. I really think this whole thing is a little bit of a straw man, you know, it's like we're all worrying about encryption. And it's like, wait, you know before it's encrypted and after it's decrypted, you can have it.

**Leo:** Right.

**Steve:** So what's the big deal of not being able to get it in transit?

**Leo:** I guess that really raises the bigger issue, which is we're all carrying in our pockets the ultimate spy device with no real means to secure it.

**Steve:** Yes. And many, many astute observers have commented that our law enforcement has never had it so good. I mean, because it was when we were going into the middle of a field under a comforter and whispering to each other that it was virtually impossible to know what was being said. The more we move into an electronic environment, the more opportunities there are for that to betray us.

**Leo:** Right. Actually, I was talking about this on MacBreak Weekly, now wearing an AI wristband that is recording everything that happens, every voice, everything, sending it to some unknown AI in the cloud, and sending me back notes, things to do, assessments. I mean, I'm basically just, you know, I'm blithely trusting this microphone.

**Steve:** Leo, we love you, but we know you gave up a long time ago.

**Leo:** I gave up a long time ago, exactly. All right. Moving on.

**Steve:** Clear my cookies? Nah, why bother. I like cookies. Okay. Okay. So whose command-and-control server is it anyway? This is an interesting story that I think everybody is sure to get a kick out of. I recall that we talked about the security research group watchTowr Labs not long ago. They're memorable not only for what they do, but because they drop the E from the "tower" of "watchTowr." So it's W-A-T-C-H-T-O-W-R. I don't know if that was a typo that stuck or what the deal is. But anyway, here's what

they posted last Wednesday about their last escapade under the title "Backdooring Your Backdoors - Another \$20 Domain, More Governments."

They wrote: "After the excitement of our .mobi research" - you know, .mobi, the top-level domain - "we were left twiddling our thumbs. As you may recall," they wrote, "in 2024 we demonstrated the impact of an unregistered domain when we subverted the TLS/SSL CA process" - you know, the Certificate Authority process - "for verifying domain ownership to give ourselves the ability to issue valid and trusted TLS certificates for any .mobi domain. This resulted in significant Internet-wide change, with Google petitioning the CAB Forum to wholly sunset the use of WHOIS for ownership validation when issuing CA-signed TLS certificates.

"As always, idle hands, idle minds. It was never going to be long until our ill-advised sense of adventure struck again."

**Leo:** I love this.

**Steve:** "And at this point, the only thing holding us back is our publishing schedule. This time, our sense of adventure struck again, in the same vein of expired and abandoned infrastructure, but with a little twist. Today, we're taking you through our adventures through what we've affectionately termed 'mass-hacking-on-autopilot.' Imagine you want to gain access to thousands of systems, but don't feel like investing the effort to identify and compromise those systems yourself, or getting your hands dirty.

"Instead, you commandeer abandoned backdoors in regularly used backdoors to effectively 'steal the spoils' of someone else's work, giving you the same access to a compromised system as the person who put in all the effort of identifying the mechanism to compromise, and performing the compromise of said system in the first place. Zero effort, same result, all for the price of a domain.

"So we've been hijacking backdoors that were reliant on now-abandoned infrastructure and/or expired domains that themselves existed inside backdoors, and we've been watching the results flood in. This hijacking allowed us to track compromised hosts as they 'reported in,' and theoretically gave us the power to commandeer and control these compromised hosts. Over 4,000 unique and live backdoors later, a number which continues to grow, we decided this research would never be finished, and it would be interesting to share the results in its current state.

"So we can report that across those 4,000 unique and live backdoors, we now have access to multiple compromised governments including those of Bangladesh, China, and Nigeria; compromised universities and higher education entities across Thailand, China, South Korea, and more; and much, much more. We've so far recorded over 300MB of logs to trawl through. As always, we're quick to remind everyone we're not the first to track hackers for fun. We no doubt won't be the last. But we've enjoyed continuing to exploit what truly appears to be a hugely underrated vulnerability class, abandoned and expired infrastructure, to basically give ourselves 'theoretical' free access to thousands of systems for the cost of a few \$20 domain names."

Now, okay. Their post goes on, and I've got a link to their post for anyone who's interested in more. But what this amounts to is that they found that some hacker gangs had allowed the domain names used by infiltrated client malware which were used to reach their command-and-control servers, those domains were allowed to expire.

**Leo:** Of course. Why would you keep them, you know. I mean...

**Steve:** Who knows why? You know? Perhaps those particular hackers are now behind bars. But for whatever reason, those domains were never renewed. This meant that the watchTowr researchers were able to re-register those previously abandoned domain names to establish their own IP for them. Then, the next time the infiltrated malware performed a DNS lookup as the first step to reestablishing contact with the malicious hacker's mothership, the IP the malware received would be the researcher's. So the watchTowr folks registered those domains and pointed the domains' IP address to their incoming connection monitor. What they found was that thousands - more than 4,000 and counting - machines scattered around the planet that had previously been infected were still, today, trying to reestablish contact with their controllers.

I'm sure that the watchTowr folks were only gathering data. But many of the incoming links were to remote web shells which would allow anyone accepting such a connection to issue commands as if they were the administrators of those remote machines. Since the wayward domains were now under their control, the watchTowr folks felt free to list 31 domains they now control. I have them in the show notes. Let's see. We've got, for example, 6634596.com.

**Leo:** It makes you wonder why they bothered registering a domain like that. I mean, why even bother?

**Steve:** Well, surprisingly, it was free. So nobody had that. And then we also have aljazeera7.com, alturks.com, caspian-pirates.org.

**Leo:** They're looking for some good branding there.

**Steve:** That's right. Csthis.com, dcvi.net, drakdandy.net. Emp3ror, with the second E of emperor...

**Leo:** Oh, it's a LEET emperor.

**Steve:** ...being a numeral 3, Emp3ror.com. Flyphoto.us. Guerilladns.com. H0ld, with the O of hold being a numeric 0, hold-up.info. H4cks, with the A being a numeral 4, H4cks.in. Hackru.info. I don't know, I'm...

**Leo:** Imhabigirl. Habi. What the...

**Steve:** Habilrig? Something dotcom. Surprisingly, Leo, that domain was available.

**Leo:** You know, what's interesting is maybe they were using these also for spoofing and other things.

**Steve:** Could have been.

**Leo:** Because you don't really need a domain name. You can just use an IP address; right?

**Steve:** Well, except that IPs can be lost. An ISP can shut down an IP.

**Leo:** That's true. So you use DNS to redirect.

**Steve:** Exactly. So you want to use DNS. And, you know, remember that, famously, remember in the early days of the podcast that we talked about systems. These were bots. They were using a calendar-based formula to predict a future domain name.

**Leo:** Oh.

**Steve:** So that they were - and that's not what we see here. But there they were just gibberish domain names. And so when the good guys got a hold of some of that malware, they would reverse-engineer the algorithm, determine what the domain name would be in the future.

**Leo:** So clever.

**Steve:** Beat the bad guys to registering that domain name, and then wait for all the bots to come there and then shut them down.

**Leo:** Wow. Wow. That's [crosstalk], yeah.

**Steve:** So, yeah, a deep history of all this. Anyway, and the list goes on and on. We've only, you know, ironwarez.info, and that's only the first half of the list. So anyway, so the point is that they're using these domains as the central, you know, communications point for their command-and-control servers. And they just let them expire. But they never told the malware, like, oops, you know, we're going to go somewhere else, if they even did. Again, we don't know why they expired. They literally, they could be in jail. They might have been locked up when the domain expired so unable to reregister it. That allowed the domain to go free. These watchTower guys grabbed it. Now all of their bots are reporting in to them.

So anyway, since they have control over those domains, they said that they obtained a monster wildcard TLS cert covering all of those domain roots and began accepting HTTPS web shell connections as they came in. So, you know, just imagine. When you think about it, this is not something we've ever talked about in all these years. How many long-forgotten and unattended systems out there are hosting old malware that gangs have moved on from and forgotten? But, you know, the bad guys don't clean up after themselves. They don't shut that stuff down. They just, you know, move on. They forget about it. So it's still out there trying, you know, like calling out in the middle of the night, trying to reach out and make contact with home base.

**Leo:** Bots. Are you there, bots?

**Steve:** That just never answers.

**Leo:** Is it mostly IRC bots these days still? Or do they have other - they must have other ways to connect to them.

**Steve:** Oh, yeah, they've got all, you know, all kinds of stuff. Actually I would be surprised if IRC was still in use because it's just so - now everything's gone TLS, and they had to get a TLS cert in order to be able to accept authenticated connections.

**Leo:** Right.

**Steve:** From all these. Because typically the malware is living off the land, so it did not bother to bring a whole big, you know, TCP/TLS Internet protocol stack with it. It's just using whatever happens to be in the OS in order to establish outbound connections for it. So it needs to have a certificate in order for it to be honored.

Okay. We're going to talk about IoT devices getting the Cyber Trust Mark after we take a break.

**Leo:** After this mark. On we go, sir.

**Steve:** Okay. So last Tuesday the U.S. government announced the launch of the U.S. Cyber Trust Mark. This is a new cybersecurity safety label for our Internet of Things consumer devices. Now, it's unclear to me whether any consumers will care or even notice. Today's IoT devices are often purchased online where any such "marks" go unseen. And with so many certifying standards bodies all weighing in with their own seals of approval, what difference is one more going to make? I remember looking in a box a while ago, I think it's when we were at a retail location, Microcenter, Lorrie and I, because our router had died on the weekend. And, you know, the box is covered with little seals and emblems and certifications and things. It's like, okay.

Anyway, but there may be a reason nonetheless to hope that the presence of such a seal may mean something to IoT companies that are seeking any edge they can get. So if changing their behavior or behavior of their products in ways that enhance the privacy and security of the users means that they qualify for yet another seal of approval, then this new FCC award may have been worth creating.

In their announcement last week, the U.S. Federal Communications Commission, our FCC, said: "IoT products can be susceptible to a range of security vulnerabilities." Uh-huh. They said: "Under this program, qualifying consumer smart products that meet robust cybersecurity standards will bear a label, including a new 'U.S. Cyber Trust Mark.'" And for anyone who's curious, I have a picture of it in the show notes. It's not particularly inspired, but okay.

And so as part of the effort, that logo will be accompanied by a QR code which users are able to scan to take them directly to an information registry, which is kind of cool, containing easy-to-understand details about the security of that specific product, you know, such as the support period and whether software patches and security updates are automatic. Which this all sounds great. So it seems like something that would be of tremendous interest at least to the audience of this podcast. But I do wonder how clued-in the typical consumer is today.

Still, the registry's information will also contain details related to changing the default password and the various steps users can take to configure the device securely. The initiative, which was announced last summer in July of 2023, so that's actually summer before last, will involve the participation of third-party cybersecurity administrators who will be in charge of evaluating product applications and authorizing use of the label. Compliance testing will be handled by accredited and independent third-party labs.

Eligible products that come under the purview of the Cyber Trust Mark program will include, you know, Internet-connected home security cameras, voice-activated shopping devices, smart appliances, fitness trackers, garage door openers, and baby monitors. But not everything, of course. The "mark" does not cover medical devices which are separately and already regulated by the U.S. Food and Drug Administration, nor motor vehicles and equipment that's already regulated by the National Highway Traffic Safety Administration (NHTSA); nor any wired devices and products used for manufacturing, industrial control, or enterprise applications. So, you know, basic consumer electronics that aren't already covered under something else.

The program does not extend to equipment added to the FCC's Covered List, the products manufactured by companies added to other lists for national security reasons (Department of Commerce's Entity List or Department of Defense's List of Chinese Military Companies) nor any banned from Federal procurement. So, again, your basic consumer electronics. But that's a huge swath of stuff that doesn't already have any coverage.

In order to apply to use the U.S. Cyber Trust Mark, manufacturers who meet the eligibility criteria must have their products tested by an accredited and FCC-recognized Cyber LAB, so that's sort of following the model of UL Labs, right, where like, where you, the maker of the equipment, submit this to UL Labs in order to get their certification. So here the FCC will recognize a Cyber LAB, which will then test it to ensure that the product meets the program's cybersecurity requirements, and then submit an application to the Cybersecurity Label Administrator with the necessary supporting documents in tow.

So for their part, last week the White House chimed in with their canned statement, saying: "The U.S. Cyber Trust Mark program allows for testing products against established cybersecurity criteria from the U.S. National Institute of Standards and Technology (NIST) via compliance testing by accredited labs, and earn the Cyber Trust Mark label. This will provide an easy way for American consumers to determine the cybersecurity of products they choose to bring into their homes."

So to me this all sounds like really a good thing. Not so much that consumers will necessarily be aware and looking for it, but that manufacturers who are in a competitive environment may be willing to change their behavior in order to obtain this. Now, I did search around the various announcement pages from both last summer and more recently. There's very clearly a lot of movement on this front because, you know, the wheels turn slowly, with various companies and individuals being selected to fill key roles. That's all been happening.

But what I was unable to find at this point was any very clear specification for the criteria NIST will be setting for the behavior of complying devices. However, we've been seeing a lot of good-sounding policies coming from NIST and CISA recently, so this is very hopeful. You know, things like, remember, requiring long lifetime support and firmware updates, and in another instance requiring consumer devices to be able to keep themselves updated and even requiring that a physical button on the device be pressed before any potentially dangerous configuration change could be applied, thus preventing remote attacks that have otherwise been possible.

So these are all really hopeful changes in the right direction. And a decade from now, once all of our first-generation systems have been retired or cycled out of service, we may see a very different terrain than we have today. And Leo, you and I will probably be around to celebrate that.

**Leo:** Episode 2000.

**Steve:** Who knows, maybe the podcast will be.

**Leo:** This is good, I think this is really good.

**Steve:** Okay. So surprisingly, there was not a lot of security news around. That was all the moderately interesting stuff I was able to find. But we have a lot left to talk about. Synchting moved to v1.29.2. What we want in software is reliability and stability, with infrequent discovery and repair of the exceedingly rare obscure bug. What we don't want are daily, weekly, or even monthly updates where we're on the receiving end of the ongoing maintenance of software that advertises itself as being feature complete and finished. As I've noted before, while I like the many features of the Notepad++ editor for Windows, its author's apparent inability to either leave it alone or get it right has become a source of continual annoyance for me. You know, if supporting his work means encouraging him to keep changing it, then I'm less inclined to do so.

Now, all of that is preamble to an event I can't recall ever experiencing. Sunday morning I was surprised by an instance of Synchting, which I have open on a side monitor so that I'm able to keep an eye on its synchronization with my other location, notified me of an available upgrade. I can't recall that ever happening before. And that's exactly what you want. The bug that was fixed by the release of v1.29.2 was obscure. The person who discovered it wrote: "By changing the contents of a synced directory, it seems that Synchting crashes when scanning a subdirectory name that contains a letter 'u' with an umlaut." Okay. The report of the problem two days ago generated some online dialog as logs were exchanged and examined. And a resolution was produced Sunday morning, two days later. You know, that's like the perfect model that you want.

And since Synchting has become a favorite for many of us - it's what Leo and I both use extensively now to keep the working files on our various platforms synchronized - I wanted to let everyone know that a tiny incremental improvement event had occurred. But I also wanted to share the observation of how refreshing it is to see a highly complex and functional open source software project, that's finished, being largely left alone because it does everything it was designed to do. And so we're not being constantly hounded to update it just because, you know, its author said, oh, look, I can, you know, we're not synchronizing dishwasher firmware. Let's add that because wouldn't that be cool. No. No, it wouldn't. We don't need that.

Okay. Last week's discussion of the persistence of unencrypted email in transit, and the fact that some 3.3 million email servers worldwide, most of them located in the United States, are still not bothering to offer a TLS certificate that would allow for email encryption, triggered a lot of feedback from our listeners. I'm going to share some of it, and we're going to talk about it because it's interesting.

Philip Pedersen said: "Steve, after your piece on the non-use of TLS for SMTP, I looked at some of the email I've received. I thought it might be small businesses that had not set up certificates, but found it to be large companies, as well. The most troublesome one I found is that TreasuryDirect.gov sends their one-time password notifications in the clear.

It also seems like organizations with multiple email servers don't all have them set up for TLS. ID.me sends the Welcome to ID.me message from a non-TLS server, although the other messages sent while setting up an account," he said, "(to log into IRS.gov) were using TLS. Regards, Phil."

So Philip's note is interesting because it hints at something I want to discuss in greater detail after I share another piece of feedback. But here's the part that's interesting. Philip wrote: "The most troublesome one I found is that TreasuryDirect.gov sends their one-time password notifications in the clear." What's tricky about diagnosing email's use of TLS-encrypted connections is that it mirrors today's web browsing, where the connecting-to server is the one that's offering to prove its identity to its caller. So in the case of email it's not the sending SMTP server that offers its TLS certificate, it's the SMTP server on the receiving end that does so.

So a sending SMTP server would always have the choice of refusing to send email to any recipient SMTP server that wasn't offering to prove its identity with a TLS certificate and encrypt their conversation and any received email with a TLS connection. But otherwise, whether or not a sending server is able to protect the email it wishes to send, is up to the receiving server. Either the sender or the receiver might elect to not send or receive messages over an unencrypted connection, but it's only the receiving server that's able to offer the use of encryption for both sides to then enjoy.

Okay. So let's now look at what Travis Hayes, another of our listeners, has to say. He said: "Hi, Steve. Enjoying this week's show, as always. Regarding the TLS encryption of email, the thought occurred to me that the reason we're where we are with unencrypted transport of email between gateways is because email from the beginning has always designed to be fault tolerant with multiple hops. Just like physical mail, if something is to be sent confidentially, it's put into an envelope rather than on a postcard for everyone handling it to read.

"This is different from the design of the relative latecomer HTTP protocol, which was designed to be point-to-point. The reason S/MIME, PGP, GPG, and the like were invented was to address that; to handle the transfer of sealed packages over a system of untrusted, unknown delivery gateways. So even if widespread adoption of TLS between gateways was achieved, I still have to be trusting that my mail host, your mail host, and any intermediate gateways are trustworthy. Even if the mail exchangers talk between themselves over TLS connections, the only way to ensure confidentiality between us is to encrypt the payload itself - and that's the piece that is missing when all those one-time six-digit PINs and 'Forgot My Password' reset URLs are being sent to me.

"Until there is some way for my bank's automated systems and me to exchange public keys so they can securely send those PINs to me, it doesn't matter if my bank's ISP and Gmail connect over TLS. I think there's some interesting things that could be done with the DKIM system. We are already digitally signing email to show it's authentic. Why are we not encrypting the message body as well? Thanks again for the show. Cheers, Travis."

So the point Travis made about email being a multipoint relaying technology is crucial because, as he noted, and as we know, TLS is only able to work with HTTP because users' web browser clients directly connect to the servers from which they wish to obtain web pages and other web application data. If a browser were to connect to any sort of intermediary server, well, we would call that a man-in-the-middle attack, which we'd go out of our way to prevent. The point is with TLS (Transport Layer Security) we receive a certificate directly from the server we wish to trust which asserts that server's identity. There is no middleman mechanism.



One reason for this is that whereas web surfing is a real-time point-to-point activity, email was never guaranteed to be immediate. These days it tends to be, but that's mostly coincidence. Email was deliberately designed to be a store-and-forward system where someone's mail message would be dropped onto an SMTP server with the address of its destination and that SMTP server would then forward their email onward toward that destination. If the receiving server was not answering at the moment, another server might be tried if the destination's DNS MX (Mail Exchange) records offered more than one, or the email would be queued for later retry delivery.

Having watched the delivery queue of my own email server when it's sending more than 15,000 pieces of email every week to those in this audience who have subscribed, I've seen that it doesn't all go through quickly or immediately. Some mail gets stuck there for a while, and then it gets accepted by the receiving end. And I know that everyone has experienced the occasional delay where someone says, "Hey, I just emailed that to you. You don't have it yet?" And then a few minutes later it shows up. This store-and-forward system was what allowed the Internet's email delivery to be extremely robust back in the early days when connectivity was far less assured, and when receiving SMTP servers might be coming on and off the Internet for whatever reason.

Things have changed dramatically since those early days. One of the things that's changed is that connectivity is now pretty much always-on, and servers are pretty much always up. But during those early days that wasn't always the case. You know, even now from time to time I need to update and reboot GRC's servers. During those times, for a few minutes, GRC's visitors will receive 404 messages about the site being down, and any remote SMTP server that's attempting to deliver mail to GRC will find that they need to queue it and retry a few minutes later. So again, the need to store and forward has not disappeared.

But as I noted in thinking about Philip's earlier note, Philip's first note, any remote SMTP server that insists upon sending email to GRC over a TLS connection, or if GRC were to insist upon only receiving email over TLS connections, then that remote server would need to ask for a TLS connection which GRC would offer, which would allow that remote server to authenticate GRC and for them to bring up an encrypted tunnel with us. However, note that although we do get encryption for privacy, the authentication is only one way. GRC offers up its TLS certificate, but the incoming connecting SMTP server does not. So it's a one-sided deal.

What this all appears to represent more than anything else is just laziness, or lack of concern, really, on the part of the industry. We talked last week about how free certificates were not easily deployed using the ACME protocol because it appeared to be myopically designed for web-only use. I'll have some feedback from listeners about that in a minute. But encryption, if that's what we want, if we want encryption, it's easily obtained. As we've often discussed, standard generic Diffie-Hellman key agreement allows any two parties to publicly negotiate a secret key which they could then use for their communication. Doesn't need a certificate for that. This would protect email in transit from passive eavesdropping.

But since Diffie-Hellman-style key agreement does not itself authenticate the endpoints - again, no certificates - this would not prevent an active attacker from intercepting email communications, taking the man-in-the-middle position, then negotiating separate keys with each endpoint on either side and being able to see everything in the clear as it passes through this intercepting tap.

But we do have a potential mechanism that would solve the entire problem if anyone really cared to because, although it's not the default case for anonymous web browsing, it is possible for both ends of a TLS connection to require the other end to provide a trusted TLS identity certificate. So simultaneous mutual authentication of TLS

connections is possible. But no one really appears to care that much, and there doesn't appear to be any movement afoot to improve email security.

We do, however, care about spam and spoofing. So those problems, notice, have been solved. SPF allows a domain to specify which SMTP servers are allowed to originate its email, and DKIM allows those SMTP servers to cryptographically sign the email they send. In both cases, DNS is used to supply the SPF record and the server's matching DKIM public key. This is done to prevent others from being able to originate spoofed email claiming to come from any source that has protected itself with these measures. But even then, it's up to the recipient to care enough to check.

I'm not sure where all this leaves us. We definitely have the tools today to bring up mutually authenticated and fully point-to-point encrypted email. But if we were to insist upon doing that, before that could happen practically, all email servers would need to have current and maintained certificates, just as all web servers do today.

And this brings us to our listeners who have arranged to do so. Leo, we're at an hour. Let's take a break, and then we're going to look at what Josh Caluette in Austin, Texas said in response to my saying, "Yeah, Let's Encrypt doesn't make that easy."

**Leo:** It's kind of, you know, and I want to hear all about this, but my attitude is email is not and was never intended to be secure. You shouldn't be using email for secure communications. Use Signal or something that's encrypted and has all of those features built in.

**Steve:** Well, and I'm sure, you know, anyone who's worked with a financial agency of some sort, like when I'm doing anything that is important, they send a link which I use on the web to then authenticate myself and log in. And then it's a web communication. It's a web session where the actual work gets done, not...

**Leo:** Then it's TLS, yeah, yeah. Email was really - it's inherently insecure.

**Steve:** And here we are using it for PINs and password recovery links.

**Leo:** I know. Sigh.

**Steve:** Because it's all we have.

**Leo:** By the way, the Patch Tuesday updates came out, 161 updates including three zero-days.

**Steve:** Ooh, 161. Oh.

**Leo:** That's more patches than they've shipped in one go in, according to Brian Krebs, since 2017.

**Steve:** Wow. Ooh, baby.

**Leo:** But it's getting more secure.

**Steve:** That's right, Leo, it's settling down. It's just like Syncthing. It's all done.

**Leo:** Oh, my god.

**Steve:** And it was that pesky umlaut over the "u."

**Leo:** Yeah, you never know.

**Steve:** Wow.

**Leo:** A zero-day umlaut.

**Steve:** Wow.

**Leo:** Ah.

**Steve:** Okay. So Josh Caluette in Austin, Texas wrote: "Hi, Steve. I was just listening to last week's podcast, and I heard you mention that let's Encrypt does not support email services. However, I've been using Let's Encrypt on my mail servers for a few years now. The certbot app has some plug-ins that make this possible even without a web server. One of the plug-ins is for nginx, which is a web server, and apache, another web server, which allow it to spin up a temporary web server for the verification process, then takes it down again.

"The other plug-in is for DNS TXT verification. There is an RFC-2136 Dynamic DNS plugin which allows for dynamically updating a DNS zone with the necessary TXT record, waiting for propagation, completing verification, and then deleting the record. This works with any servers that support and are configured to allow Dynamic DNS updates securely using private keys.

"There's a similar plug-in which I use specifically for Cloudflare. It does the same thing, but it works with the Cloudflare API to dynamically update the DNS zone with the correct TXT record. Once the certificate has been generated or renewed, it can be used in the config of anything that accepts certificate private/public keys. Because the file names do not change, you can easily configure services to point to the Let's Encrypt managed files and then configure certbot with a post-script to restart the necessary services in order to begin using the new certificate. I've been using this for the past couple of years, and it has worked great, with no intervention.

"I have some monitors set up that monitor all of the certs used by services and alerts me if any of them get within 28 days of expiration, as that indicates a problem, since they should be renewed by or before reaching the 30-day mark. But anytime there's been a fault, it's been due to my own errors - firewall changes, bad configuration changes, et cetera.

"Thanks for all you do. I look forward to the podcast during my two-days-a-week commute to and from work."

Okay, now, I think Josh's note serves to illustrate two things perfectly: Yes, it's possible; and no, it's neither automatic nor easy. And, not surprisingly, many of our listeners who are technically sophisticated and capable of rolling their own kludges have similarly done so. And a kludge it is. That's the proper term for arranging to create a temporary web server to satisfy a port 80-only certificate-issuing service, or dynamically editing DNS and waiting for propagation, then copy the resulting certificate around and restart all dependent services nightly so that the updated certificates are recognized.

**Leo:** Wow.

**Steve:** That's the very...

**Leo:** But you know what? That's not the hardest thing about running your own email server, either, okay. But admittedly you're pretty sophisticated, yeah.

**Steve:** Yeah. It's the very definition of a kludge, as I mentioned last week. Now, I fully intend to do something similar, I'll have no choice, if the lifetime for all certificates are forced to drop below one year.

**Leo:** Yeah.

**Steve:** Given that long certificate lifetimes appear to be an entirely made-up problem, the more I've thought about this, the more it seems that web browser certificates should be members of a separate elite class, if that's what they want. Let them expire every six days, so long as anyone offering the ACME protocol will keep them all fresh. But then leave everything else alone. Let non-web servers use good old reasonable three-year life certificates for, you know, our Internet appliances, email servers, and other things. Don't force this nonsense down everyone's else's throat. Or allow those of us who wish to obtain an identity-asserting certificate - for which we're paying good money - to decide for ourselves how long that certificate should last. Obviously, every time I talk about this I get myself all worked up. This just really rubs me.

**Leo:** Yeah.

**Steve:** So let's change the subject. Doug Curtis in Waukesha, Wisconsin said: "Steve, thanks so much for your overview of the current state of the art on AI. It prompted me to use ChatGPT to get some recommendations for more sci-fi books. I've really enjoyed some of the recommendations that you have made over the years in various Security Now! episodes.

"I received a gift for Christmas of several credits toward Audible, so I wanted something new to listen to. I started by asking ChatGPT about two of my favorite sci-fi series, the Bobiverse and the Giants series. And then I asked it, based on those two series, if it could make recommendations based on my preferences. It made quite a few. I'm starting with something called the Murderbot Diaries by Martha Wells."

---

**Leo:** Oh, yeah. Stacey Higginbotham loves those, yeah.

**Steve:** He said: "So far, and a few chapters into the first book of the series, I'm liking it very much. The next book that I'm going to try after this one is one called 'Old Man's War' by John Scalzi. I've read another of his books, called 'Starter Villain,' which was very interesting. Thanks again for all you and Leo do with the Security Now! podcast and for your SpinRite program. I have a license and have used it several times. Regards, Doug."

**Leo:** Yay.

**Steve:** Okay. So I've not looked at the Murderbot Diaries, but I've definitely heard of them, and other of our listeners have recommended them to me. And the same goes for John Scalzi's "Old Man's War" novel. It's been recommended, too. Taking Doug's clue of asking ChatGPT for some recommendations, I logged into my own ChatGPT account, selected the full-strength o1 model, and asked the following question: "I enjoy reading science fiction novels, and my favorite author is Peter F. Hamilton. I also enjoyed the Silver Ships series, and Ryk Brown's Frontiers Saga series. Can you recommend other authors whose science fiction novels may be similar?"

Now, this gave it some pause while it worked through what I was asking. The o1 model shows you the various phases of its work as it's going along. After cogitating for a full 10 seconds, it replied: "If you're a fan of large-scale space opera (like Peter F. Hamilton's Commonwealth or Night's Dawn universes) and long-running series with plenty of interstellar adventure (like S.H. Jucha's Silver Ships or Ryk Brown's Frontiers Saga)" - which of course it apparently understood the natures of because it correctly characterized them. It wrote: "Here are some authors/series that capture a similar feel."

First recommendation: Alastair Reynolds. He said - it. He. It. It recommended. So Alastair Reynolds recommended - he said, it rec - I can't, I'm having a problem with my pronouns.

**Leo:** It's hard, isn't it. Not to personify it, it's really hard.

**Steve:** It is. "Recommended starting point: 'Revelation Space.'" Which I should say I have read and very much enjoyed.

**Leo:** Good.

**Steve:** It wrote: " Why you might enjoy it: Reynolds is known for sweeping, galaxy-spanning storylines that blend astrophysical plausibility with far-future imagination. If you like Hamilton's big ideas, Reynolds' Revelation Space universe offers a similar sense of scale and mystery." And I agree. They were a little dark, but they were definitely captivating.

Second recommendation: Neal Asher. It said: "Recommended starting point: 'Gridlinked' or 'The Skinner,' both set in the Polity universe." As for why I might enjoy it, it wrote: "Asher's Polity novels deliver gritty, action-packed stories in a post-scarcity future run by powerful AIs. The cast includes enhanced humans, alien threats, and a lot of big tech ideas, echoing the grand technology vistas in Hamilton's work."

Third recommendation: James S.A. Corey. No surprise. "Recommended starting point: 'Leviathan Wakes' (Book 1 of The Expanse)."

**Leo:** Oh, yeah, of course.

**Steve:** "Why you might enjoy it: The Expanse is a multi-volume series known for its realistic(ish) physics, gritty political drama, and deep space exploration. While it starts in our own solar system rather than spanning galaxies, it still provides epic storytelling and intense space conflicts."

And then I have to say my eyebrows rose. Number four, Jack Campbell. "Recommended starting point: 'Dauntless' (Book 1 of The Lost Fleet)," which was one of my earliest recommendations for our listeners.

**Leo:** I remember, yeah, yeah.

**Steve:** It said: "Why you might enjoy it."

**Leo:** In fact, you recommended all of these books except for the Asher; right? Yeah.

**Steve:** Yes, yes. "Campbell's Lost Fleet series focuses on large-scale fleet engagements, strategy, and leadership under pressure. Perfect if you like space opera with a strong military twist, intricate battles, and a big overarching plot." And again, I'm just - I'm astonished by this response.

Fifth, David Weber. "Recommended starting point."

**Leo:** Oh, my god.

**Steve:** "'On Basilisk Station' (Book 1 of the Honor Harrington series)."

**Leo:** Yeah, I think you might like that, Steve.

**Steve:** Uh-huh. Maybe just a bit. Read it twice. And I actually read the Lost Fleet series several times. And it said: "Why you might enjoy it: While the Honorverse is definitely military sci-fi, it features expansive world-building across numerous star systems, detailed technology, and a strong continuity across a large number of books, much like Hamilton's broad universe-building."

**Leo:** Huh.

**Steve:** Number six, Elizabeth Moon. "Recommended starting point: 'Trading in Danger' (Book 1 of the Vatta's War series). Why you might enjoy it: Moon's style blends space opera, military sci-fi, and strong character development. The Vatta's War series

follows Kylara Vatta and her family in an interstellar trading empire, so you get both epic space battles and corporate/political intrigue."

Seventh: Glynn Stewart. "Recommended starting point: 'Starship's Mage' (Book 1 of the Starship's Mage series, or Duchy of Terra series. You might like it because Stewart's series offers combined space opera tropes - like starship crews, alien empires, and grand-scale warfare - with a smooth, action-driven narrative. They're fast-paced, entertaining reads that grow into multi-book arcs."

Finally, number eight, Michael Cobley. "Recommended starting point: 'Seeds of Earth' (Book 1 of the Humanity's Fire trilogy). You might enjoy it because Cobley's Humanity's Fire features multiple human colonies struggling after Earth's near-destruction, alien civilizations, and galaxy-scale conflict. If you enjoy multi-threaded plots and epic storytelling, this trilogy is worth a look." I'm reading what it wrote. I'm just astonished.

And in the "More to Consider" category at the end, there's John Scalzi, known for his blend of humor and military sci-fi, especially Old Man's War series. Also, Lois McMaster Bujold. "The Vorkosigan Saga is character-driven, swashbuckling space opera with great political intrigue." Ian Douglas and Joel Shepherd.

Anyway, the summary at the end: "If you love the grand, interstellar scale of Peter F. Hamilton's books and the long, connected story arcs found in Silver Ships and The Frontiers Saga, these authors deliver similar epic space-operatic adventures. Whether you lean more toward military sci-fi or more traditional space opera, there's sure to be a new universe here to explore."

**Leo:** That's hysterical.

**Steve:** Unbelievable.

**Leo:** Now, had you read all of these, most of these? I know the first three or four you knew.

**Steve:** No. I'd read Alistair Reynolds. I don't think I ever mentioned it. But of course Jack Campbell and the Lost Fleet, and David Weber with the Honorverse and Honor Harrington, yes, of course.

**Leo:** Yeah, yeah, of course. And it could be cribbing from your show notes, to be honest with you.

**Steve:** Could be. And I did - yes. And it occurred to me, and I did publish a Steve's sci-fi reading guide PDF that does have the earlier works, the Lost Fleet and the Honorverse stuff. So wow. And Leo, I have a new author.

**Leo:** Oh, good.

**Steve:** I decided I had heard a lot of Neal Asher mentioned. I had never read any of his books. I've already started, and I cannot put it down.

**Leo:** Oh, good.

**Steve:** It really looks - and the good news is [crosstalk] a lot.

**Leo:** Which one are you reading, "Gridlinked" or "The Skinner?" "Gridlinked" or "The Skinner"?

**Steve:** "Gridlinked."

**Leo:** "Gridlinked."

**Steve:** "Gridlinked." It is just - I just - and I have to tell you, Leo, a couple weeks ago I was - I had finished this latest Hamilton workout. And I thought, I need something - I need something simple.

**Leo:** Yeah.

**Steve:** I overdid it in the simplicity category.

**Leo:** You went too far.

**Steve:** The book, it was called "Artifact," and it began, I kid you not, the book started, "As it dropped out of orbit, the alien starship Zigawatt..."

**Leo:** Oh, never mind.

**Steve:** And at that point...

**Leo:** Bye-bye.

**Steve:** I just - I should have stopped.

**Leo:** Bye-bye.

**Steve:** You actually called your alien starship Zigawatt? No. No. Anyway, I've been saved by Neal Asher.

**Leo:** I'm going to have - I've never heard of him. I'm going to have to look that up.



**Steve:** I had heard of him. And I thought, it occurred to me that it's somewhat fitting that after finishing the first novel in Peter Hamilton's newest two-book series, I plowed into the research to understand how ChatGPT and similar Large Language Models operate. And after having done so, that technology has just recommended how I might best resume my previously interrupted work to return to science fiction. I believe that's what's known as closure. So, yeah.

**Leo:** Yes, full circle, yeah.

**Steve:** This "Gridlinked" novel, whoa. I mean, it is exactly - it's just - I just want really good writing, more than anything else; you know? Not Zigawatt, not so much. But this is like, whoa. The author makes you work a little bit to understand the meaning of new terms. And then it's like, oh, I know where that came from.

**Leo:** Ah, interesting.

**Steve:** And anyway, it's good. It's good. Okay. Bob said: "Hi, Steve. I want to supply some feedback to your last show regarding auto updates of hardware. I don't agree with your comment that enterprise-level network security appliances, firewalls, routers, and switches should be set up with automatic updates. History has shown that automatic updates can cause devastating outages for businesses. I find it doubtful that you would turn on automatic updates on any of your systems."

Uh, okay, well, he's got me there, yeah. I'm not at all certain that I would take my own advice in that regard. But he continues: "Maybe the point here should be if a person's company does not have the staff, knowledge, experience, or money to have test systems that can be used to install updates and confirm that they're working as expected, then and only then using automatic updates makes sense, since at least that way they would be protected from unpatched vulnerabilities. But again, they would probably be better served with a managed service partner taking care of their systems for them."

Okay, and of course now that's meaning that smaller enterprises should perhaps outsource the responsibility for managing the infrastructure which on the one hand they need because you need to have a network, and it needs to be connected to the Internet these days; but which on the other hand they don't have the staff focus or care to maintain for themselves. So I think Bob makes a good point, even though we have seen the MSP route go very wrong when the MSP's network was compromised, which allowed bad guys to get into the networks of all of their clients.

Anyway, Bob continues: "I retired," he wrote, "from a multinational transaction processing company. After a security breach they implemented tightened security procedures that I am surprised more companies don't. This company has more than 50,000 employees." He said: " We used network segmentation, and the office network was not able to directly connect to the transaction processing network without going through a Bastion Server which was fortified, locked down, and had separate two-factor authentication. All new servers had to have a defined owner contact and business unit owner. All firewall rules had to be justified, and these rules needed to be reviewed by the business unit owner quarterly to ensure that the rules were still needed. All hardware and software had to be supported by the manufacturer.

"Patches needed to be installed within two weeks, sooner if the issue was critical, allowing time for testing, production beta testing, and full rollout. We had redundant data centers, so we'd first install into the production data center. And if the updates caused

issues we'd fail over to the unpatched backup systems." These guys were serious. But again, a 50,000 employee, some sort of transaction processing center, I mean, that's a big global enterprise that is - and we don't know who these people are, but yeah. He said: "All software being run on any systems needed to be whitelisted." Meaning you can't even run anything that isn't permitted. So it's not a blacklisting system, it's whitelisting. Meaning deny all, permit specifics. "Any exceptions," he said, "needed to be reviewed and approved. No personal devices could be used on any networks." Wow.

He says: "I won't even get into the DDOS and web app firewalling we used." He says: "My point is security is tough, and employees hate it." He said: "I know, because they kept complaining to me how much harder their jobs were once we implemented the clearly required security measures. My comment back to them was that they were being paid very well, and if we were breached they likely wouldn't have a job because clients would drop us and move to a competitor." And he finished: "Love your show. Happy New Year. Bob."

So Bob's note is a perfect case in point for the tradeoff of convenience versus security. And imagine the extra cost to this organization of doing all this. This isn't free, either.

**Leo:** Oh, man. Yeah, plus the cost of a breach, either; right?

**Steve:** Exactly. And the reputation damage, that takes a long time to amortize out. And, you know, you might imagine the sour comments of an employee who relocates from a company with very little security and lax useless controls, to one with strong and truly useful security. Such an employee might well be grouching about how they didn't need to do all this or that at their last company. Right.

And finally our listener Patrick Beemer, who runs a 15-year-old Managed Service Provider himself, you know, an MSP, shares some background on SonicWall. Patrick wrote: "Hey, Steve. I'm listening to your commentary about SonicWall exploits." Remember we talked about them last week, about how so many of them, after four months from a critical patch being made available, 10% still had not been patched, and how many apparent vulnerabilities on the public Internet remained.

He said: "I'm listening to your commentary about SonicWall exploits, and I wanted to provide some additional thoughts about why over 10% of the installed base is still vulnerable to an exploit from August of 2024." He says: "I run a 15-year-old Managed Service Provider and have been a SonicWall partner from the beginning. SonicWall firewalls were mandatory for all our clients." He says: "(We're slowly moving our clients away from 'big iron' at the edge for reasons that are not relevant to SonicWall as a company or this message)." He said: "SonicWall is a very popular firewall for small businesses and MSPs. These aren't large companies with IT departments, but are typically orgs with 10-15 staff that rely on an MSP or maybe a" - I love this term, I've never seen it before, Leo - "a solopreneur." They "rely on an MSP or maybe a solopreneur to support them." You know, a one-man tech firm, small.

He said: "Worse, many companies this size choose not to maintain a relationship with a support partner. These firewalls are just sitting there, waiting to be exploited. And there's A LOT of them," he said, all caps.

"Secondarily, Leo asked why SonicWall doesn't just push the firmware updates. Two reasons. First, concern about impact, responsibility, and liability. Sitting at the edge of a business, a firewall with a bad update immediately becomes a hair-on-fire emergency. As an MSP, I wouldn't want SonicWall pushing updates at my clients' firewalls. That's not their job. The risk here for SonicWall is too great. The other reason is that SonicWall sells

features. And the feature that enables cloud-based, scheduled firmware updates costs extra, a cost that many budget-conscious businesses are unwilling to invest in." He said: "(We make it mandatory)." He said: "I hope that provides a little context about why this is still a thing.

"Finally, I want to take a moment to thank you and Leo for the expert guidance I've received over the years. I've been following Leo since the '90s. I started using ShieldsUP!..."

**Leo:** Oh, that's who's been behind me. I was wondering who was behind me.

**Steve:** He's been following you since the '90s. He said: "I started using ShieldsUP! almost as soon as it came out, and have been following you both ever since. Though it wasn't until I got my CISSP in 2019 and needed a reliable source of CPEs that I started listening regularly. And I'm also a member of Club TWiT."

**Leo:** Yay.

**Steve:** "The information you provide each week keeps me informed and makes my job easier. Thank you. Cheers, Patrick Beemer."

**Leo:** We need to - that gives me an idea for a slogan for joining Club TWiT is "It's cheaper than a firmware update feature," or something like that.

**Steve:** Yeah.

**Leo:** Maybe [crosstalk].

**Steve:** There were lyrics to a song, or maybe it was a title, "It's cheaper to keep her."

**Leo:** Cheaper to keep her.

**Steve:** Which I never got out of my head, yeah.

**Leo:** Seven bucks a month, it's cheaper to keep her.

**Steve:** Well, I thought Patrick's information was great. At first I thought I had spotted a contradiction since he noted the potentially catastrophic danger that automatic updates posed.

**Leo:** Yeah, but he makes them mandatory.

**Steve:** Well, then he later noted that automatic updates were actually available for an extra fee.

**Leo:** That's the problem.

**Steve:** So which is it? Either they're a danger, or they're a benefit. But the way out of this conundrum is that SonicWall makes their customers pay for the privilege of these automatic updates, doubtless with an ongoing subscription. And I'm sure that part of that agreement with SonicWall is that keeping one's firewall updated is a good thing, thus the reason for offering the service in the first place.

**Leo:** Good point.

**Steve:** But if something happens as a result, we did the best we could; and, after all, you paid us to do this for you because it's what you asked for.

**Leo:** Oh, that's a good point. Lets them off the hook a little.

**Steve:** So it sort of, you know, it takes the danger level down a bit.

**Leo:** Yeah, yeah, yeah.

**Steve:** Okay. So we're right on schedule. We're at an hour 34. We are now going to roll up the sleeves and dig in.

**Leo:** Yes.

**Steve:** So we'll take a break. We have one left, which I'll break in the middle of the balance of this because people are going to need to catch their breath, I think.

**Leo:** It'll be a good break, yeah.

**Steve:** It's going to be - where we're headed is not for the faint.

**Leo:** You can run out and get a Werners or something to stimulate the brain. All right, wait a minute, let me get my thinking cap on. I'm ready. Let's talk about HOTP and TOTP, Steve.

**Steve:** Okay. As I mentioned at the top, today's topic was inspired by feedback from one of our listeners. Max Feinleib sent two notes, two weeks apart. I collected his two questions, which I initially started out answering as part of our regular listener feedback. But as my answer's length grew, I realized that not only had we somehow never, at any point in our 1,007 prior episodes, talked in detail about something that probably every

one of us is using, but I believed that the details of the technology that's going on would be something everyone would enjoy thinking about because there's more to it than you might think.

Okay. So to get us started, here are the two pieces of feedback Max provided. He said first: "Hi, Steve. I've been noticing lately that the six-digit codes I get for two-factor authentication almost always seem to include one or more repeated digits. Of course, you'd expect some repeated digits. Nearly 85% of six-digit numbers have six unique digits. However, my sense is that there are more repeats than there should be. I see a lot of codes that only use three or four unique digits, like, say, 906090 or 131327. It feels like the codes are being biased toward numbers with repeating patterns to make them easier to type.

"Have you, or any other listeners, observed this? If two-factor authentication codes are truly being dumbed down in this way, how much of a concern is that? Maybe it's not a big deal because the 30-second rotation makes brute-forcing two-factor authentication codes quite difficult. To note: I use Cisco Duo for my personal accounts and Microsoft Authenticator for my work accounts. Both apps seem to give me these dumbed-down codes. Thanks, Max."

Then, two weeks later: "Hi, Steve. I wanted to follow up on this question. Over the past several weeks since I sent you this, I've continued to note my two-factor authentication codes. I've continued to get much below 15% of my codes with six unique digits, and it's far more common to have two repeats or a tripled digit. My mom has been doing the same, and she's only told me about two occasions when she got a code with six unique digits. So I still believe that two-factor authentication codes are being dumbed down for easy typing. Would love to hear if you can find anything on this."

**Leo:** This is really interesting. I'm looking at my 2FAS app and looking at all the codes, and he's right.

**Steve:** Yes.

**Leo:** I don't - I see very few, I don't think I have any with six unique digits. He's saying that 85% of all numbers should not have a repeat, or should have a repeat?

**Steve:** Yeah. It turns out, and I think either I did or he did, somebody, I think maybe I did, I asked ChatGPT, which I'm still having fun with, and it stunned me again. It perfectly explained where that 15% came from. It explained that for the first digit, you have any one of nine possibilities. Then for the second digit...

**Leo:** Oh, yeah, eight.

**Steve:** ...any one of eight possibilities, then any one of seven, then anyone of five and so forth. And when you do the math, multiply it out, and divide it by a million possibilities, you know, from zero zero zero zero zero zero to nine nine nine nine nine nine, that's 15%.

**Leo:** Wow. That makes sense.

**Steve:** It's like 15.21 or something like that.

**Leo:** Yeah, okay, okay.

**Steve:** Yeah. So you can do the math. Okay. So before we examine Max's observation, his question, as I said, points out that in none of our previous 1007 podcasts have we ever taken the time to examine exactly how and where these time-varying digits are generated. Since that bears upon Max's observation, as the saying goes, no time like the present. But even more, this provides the perfect setup for one of our theoretically pure deep dives into fundamental computer architecture and technology. And buckle up because there's more here than you might imagine.

**Leo:** Okay.

**Steve:** Even the gurus among us who know all this, yeah, maybe give you something to think about. TOTP, which is the abbreviation for the algorithm that all time-based authentication uses, stands for "Time-Based One-Time Password" algorithm. It was standardized and specified in RFC 6238 back in 2011. It builds upon HOTP, the "HMAC-Based One-Time Password" algorithm which was standardized and specified by RFC 4226 in 2005. We positively know that these standards are what everyone must be uniformly using everywhere, otherwise there would not be, and could not be, the universal agreement we obviously have about the proper six-digit code to use at any point in time.

So that's established. These are the governing standards and specifications. So this allows us to dispassionately examine those two RFCs to see what they say, knowing that they must be operative. Of the two, the only one that matters is the earlier HOTP since that's the standard that's used to generate the digit sequence, with TOTP just being used to feed a new time-based value into HOTP every 30 seconds.

HMAC (HMAC) stands for Hash-based Message Authentication Code, where the HOTP standard uses the long-proven, well known to be cryptographically secure HMAC SHA-1 hash algorithm. As we've discussed many times on this podcast, any cryptographic hash function, such as SHA-1 in this case, takes an input plaintext of any length and "digests" it into a fixed-length output. That's all any hash function does. So we can imagine that we are wanting to somehow hash the current time of day and date to produce and then display a random-ish result. The problem is, if that's all we did, everyone's authenticator would be producing the same random-ish result all the time. What we need to do is introduce the idea of a secret key so that we can create a collection of these time-varying random-ish outputs.

Once again, our cryptographic toolkit provides a perfect tool, known as the HMAC. The long-established and well-proven HMAC algorithm uses any cryptographic hash at its heart, but it also adds the provision of a key. So it essentially takes an unkeyed and unkeyable generic hash function and turns it into a family of hash functions where the particular hash function performed is determined by the HMAC's key.

So now we have the basis for what we need. A remote server randomly generates a secret key to be used for authentication for a specific user. It converts that secret key into a QR code and presents it to the user as part of their identity sign-up. The user's authentication app scans the QR code to capture and retain that key. And the remote server stores that key with their account.

Subsequently, at any point in the future, with each endpoint having the same shared secret to key their respective HMAC functions, they're each able to "HMAC" the current time of day and date which will result in an identical output. And since the output will only be identical if both HMACs are identically keyed, this allows the re-authenticating user to prove that they still have the previously shared secret key without ever divulging what it is. And since this correct output is based upon the time of day and date with 30 seconds of granularity, anyone who might arrange to intercept or capture the authenticating conversation will not have obtained anything that they can use in the future since they won't ever have the secret key. So we have an extremely elegant solution that is working well for us today.

I wanted to first establish this foundation for those who may not have been with us from the start so that we're not missing any critical pieces for what comes next. At the heart of every HMAC lies a hash function. And in the case of the TOTP and HOTP functions, which were standardized back in 2005, that hash function is the venerable SHA-1. The SHA-1 hash takes whatever is fed into it and hashes that into a fixed-size, 20-byte, 160-bit hash digest.

What we know about any cryptographically secure hash is that the bits produced by this hash are all, every single one of them, completely pseudorandom. The SHA-1 hash has been in use for decades, and its bits have never shown any discernible pattern that would weaken it. The only reason SHA-1 has been deprecated over time is that, these days, the world has much more processing power available for hacking and cracking than it once did. So we'd prefer to have more bits of hash output just for the sake of more is better, and it makes us feel more secure. Consequently, the world has moved to the newer family of SHA-2 hashes, typically using SHA-256 to give us 256 bits or 32 bytes of hashed output.

Okay. Now, I can hear some of our more informed listeners grumble that this old SHA-1 hash was found to have some weaknesses. That's true. But none of those ever related to the use of the hash for the generation of high-quality pseudorandom data. There were some so-called pre-imaging attacks against SHA where it was being used to generate a cryptographically secure signature for a document. We never want to be able to manipulate the input of a signature's hash so that we're able to design a modified document that winds up having the same hash, and thus signature, as the target document. That would completely break the guarantee that document signing provides. Over time, SHA-1 was found to have some weaknesses there.

As junior cryptographers, the important takeaway lesson for us is that just saying "SHA-1 is broken" is a simplification that is untrue. The "brokenness" of a cryptographic function almost always depends upon how that function is being used. And SHA-1 remains a perfectly good and cryptographically strong pseudorandom number generator. For this application as a pseudorandom number generator, it needs no upgrade or replacement. This is why the entire industry has remained standardized upon it, even today in 2025.

Okay. So with 30-second granularity, the UTC time - as in the current time and date, along with a secret key, is fed into this SHA-1 HMAC which converts it into a cryptographically strong pseudorandom set of 160 bits, which is 20 bytes. So we have what is essentially 160 pseudorandom bits. This can be viewed as a single very, very large decimal number ranging from 0 to 2 raised to the power of 160, which is - okay. Now, it's in the show notes. Leo put it on the screen. Thank you, Leo. I cannot begin to pronounce this. It is 1,461,501,637,330,902,918,203,684,832,716,283,019,655,932,542,976.

**Leo:** Wow.

**Steve:** That's the number.

**Leo:** Ask GPT how you say that in English.

**Steve:** Oh, you probably could.

**Leo:** I bet I could, yeah.

**Steve:** It is a 49-digit decimal number. So that gives you a sense for the size of - that's the number of combinations that you can have of 160 binary bits. So, I mean, this is why binary and bit length is so powerful. There's only 160 binary bits, but you get that many combinations of them.

Okay. So now let's explore, because this is the output of the HMAC, 160 pseudorandom bits...

**Leo:** Do you want to know?

**Steve:** Okay.

**Leo:** Let me put this up on the screen. One quattuordecillion, 461 tredecillion, 501 duodecillion, 637 undecillion, 330 decillion, 902 nonillion, 918 octillion, 203 septillion, 684 sextillion, 832 quintillion, 716 quadrillion, 283 trillion, 19 billion, 655 million, 932,000, 542,000, and 976.

**Steve:** Wow.

**Leo:** It's an extremely large number, says Perplexity.ai.

**Steve:** And what's interesting is it got the digit count wrong.

**Leo:** Oh, did it?

**Steve:** Yeah.

**Leo:** Oh, yeah, it says it's 51 digits. That's not right.

**Steve:** It's 49 digits.

**Leo:** Huh.



**Steve:** Isn't that interesting.

**Leo:** Huh. I wonder if I - no, I think I pasted the right thing in.

**Steve:** One, yeah, it starts off, yeah, I mean, I'm looking at it, and it is exactly right.

**Leo:** Well, that's - this is the kind of weird thing. This isn't ChatGPT. I can try, let me try, well, go on with the show. I could spend a lot of time on this one. I'll do it on ChatGPT, see what it says.

**Steve:** Yes. I counted the groups of three between commas. There's 16 groups of three, so that's 48. Plus the one in front is 49 digits. So that is the kind of thing that these things get wrong.

**Leo:** Little weird things like that, yeah.

**Steve:** They're not math machines. They're generalizers. Yeah. Okay. So I need your attention on this, Leo, because you're going to love this.

**Leo:** Okay.

**Steve:** Okay. So let's explore because this is the output from the HMAC. The HOTP HMAC is these 160 pseudorandom bits. So now let's explore the various ways that we might go about converting this humungous 160-bit, 49-decimal digit, or 20-byte SHA-1 based HMAC output into those six digits that we want our fancy authenticator to produce. Thinking of this as a very large and long binary number, let's first say that we wanted to extract digits only ranging from 0 to 7, which is to say any one of eight possible values; right? Zero through seven, eight values.

One approach would be to shift the entire large number three bits to the right. In binary math, shifting a binary number to the right divides its value by two. And the bit that's shifted off the right end is the remainder of the division by two. So if we shift a large value three bits to the right, that divides it by eight because it's divided by two, three times. And the three bits that would be shifted off the right end would be the remainder of the division by eight. That would give us a binary number ranging from 0 to 7. Those three bits give us a binary number ranging from 0 to 7, and when converted to decimal, a single digit between 0 and 7.

So by dividing the massive number by eight, we're able to "extract" a digit ranging from 0 to 7. And we could do this again and again, as many times as we need, to extract as many digits from the large number as we need. But we do not live in an octal world, presumably because we do not have eight fingers and toes. We have 10 fingers and toes, so we count in decimal, with a 10-digit alphabet ranging from 0 through 9. And it's a 10-digit alphabet that we need our TOTP and HOTP to produce.

So here's the coolest thing: Since our fingers- and toes-friendly authenticator wants to produce one-time passcodes containing all 10 digits ranging from 0 to 9, instead of dividing the very large number by eight, we divide it by 10. Dividing any large number by 10 will give us a remainder ranging from 0 to 9. The solution is clean, simple, and

elegant. If it had been left to me to design the digit extraction algorithm for the HOTP algorithm, I would have done exactly that. I would have simply successively performed a very long division of that very large 160-bit number by 10, taking the remainder from each division, which would have resulted in an extremely uniformly distributed digit range from 0 to 9. And that simple long division could have been repeated as many times as needed to successively extract as many pseudorandomly determined digits as needed.

And if we generalize this a bit, just for the sake of cool math and theoretical computer science, what's so cool about this approach is that it is wonderfully generic. If the size of one's alphabet happens to be exactly some power of two, then dividing any binary number by that is as simple as shifting the binary bits of that number to the right and grabbing the bits that fall off the end. They form the choice for the item extracted from the large number.

But having a practical alphabet size that's exactly some exact power of 2 would mostly be coincidence. The usual case is that the size of the alphabet is whatever it is. If we want to extract decimal digits, we divide by 10. If we wanted to extract evenly distributed English alphabetic characters, we could perform long division by 26, then map the resulting remainder, which would range from 0 to 25 to the letters of the alphabet "A" through "Z." Or if we wanted both upper and lowercase alphabetic characters, we'd divide by 52 to get a remainder that could be mapped to both lower and uppercase alphabet. And if we wanted upper and lowercase plus decimal digits, we'd divide by 62, and so on.

This is exactly what I did with the design of the Perfect Paper Passwords system which we talked about during Security Now! Episode 115 which, Leo, you and I recorded...

**Leo:** A long time ago.

**Steve:** ...on October 25th of 2007. The Perfect Paper Passwords system successively performs long division of a very long number by the size of the alphabet the user wishes to use. This generates successive division remainders of exactly the alphabet size which is used to enumerate successive items of the alphabet. So in the case of something like HOTP, this clean and simple approach of the long division of the entire 160-bit SHA-1 number by 10 would allow any number of decimal digits to be extracted from that very long value to satisfy the need for a maximum quality pseudorandom decimal number having any number of digits. Boy, that brings back some memories, Leo.

**Leo:** Look at that. Look at that. But you can read this whole thing, that you're doing the same thing. You can read it all there; yeah?

**Steve:** Yup.

**Leo:** That's really - that's super cool.

**Steve:** Isn't that? Just it's so slick.

**Leo:** Yeah.

**Steve:** But I said that's what I would have done if I'd been given the task.

**Leo:** What did they do?

**Steve:** And as I said, it's what I did do, back in 2007.

**Leo:** Right.

**Steve:** But the group who designed the HOTP - uh-huh, you're right - algorithm, they didn't ask me, and that's not what they chose to do two years earlier in 2005. Looking at what they chose to do makes me want to scratch my head. The only rationale I can come up with for what they designed - the term, being kind, would be "ad hoc" - was that it was good enough, and that perhaps they didn't trust coders who would be implementing their standard to be able to divide a long binary number by 10.

**Leo:** Is it computationally expensive? No.

**Steve:** No. It's elegant, and it's beautiful. I mean, and actually the code in assembler, which, you know, is where I wrote it, is just wonderful. But you can, you know, you can do it in anything because you're just shifting bytes along.

**Leo:** Yeah, yeah.

**Steve:** So they went way out of their way to avoid that.

**Leo:** Oh, dear.

**Steve:** Okay. Okay. So I wanted to first explain, as I just have, the cryptographically optimal way of solving this simple problem of computer science so that everyone would have a reference point against which to judge what actually transpired. Get a load of the universal HOTP algorithm that we all wound up with for better, for worse. And Leo, we will continue after our final break.

**Leo:** Oh, you're mean. That's a tease. Wow. Wow.

**Steve:** You won't believe it. You won't.

**Leo:** So when you do it your way, it's going to be completely uniform in the distribution; right? It's not going to favor any digit. It's just it's random, and it's going to be uniform in its distribution doing it your way.

**Steve:** What's so significant about my way, and we'll actually visit this explicitly later because we're going to look at how, like, how bad their compromise makes things is that I'm always using all the bits.

**Leo:** Right. That's important.

**Steve:** Well, if you want the cryptographically perfect solution, that's how you do it.

**Leo:** Right.

**Steve:** That's not what they did.

**Leo:** You don't throw out entropy.

**Steve:** Oh, boy, did they. Oh. You come limping across home plate with barely enough entropy.

**Leo:** Oh, my god. So in other words, our correspondent was right to say, hey, something seems fishy.

**Steve:** Well, we'll get to there, too.

**Leo:** Oh, good. All right.

**Steve:** We'll answer that question.

**Leo:** Okay. This is good. I hope you're following along. I'm only kind of sort of following along. But I'm getting the general gist of it. I'm surprised that Advent of Code has not had this as a challenge. I think they actually have, come to think of it, to do your own hashing algorithm. Anyway, hey, one more thing before we get back to Steve and some number crunching, more number crunching. I would be greatly appreciative if you would go over to our website, [TWiT.tv/survey](https://TWiT.tv/survey), and fill in our survey. It should only take you a few minutes. It's the one thing we do once a year to try to get to know our audience as a whole.

We're not collecting information about you individually. But we like to know more about our audience, what they're interested in, what their occupations are, their ages, things like that, for two reasons. It helps us design programming that's better suited to you. But it also helps us sell advertising because advertisers, they always want to know all about you. And we don't want to tell them anything about you. But we like to be able to say things like, oh, you know, 75% of our audience are IT decision-makers. That's useful. So fill it out, if you will. It really helps us. You've got maybe a week more before we take it offline. [TWiT.tv/survey](https://TWiT.tv/survey). And thanks in advance. I appreciate it.

Now, get your propeller heads back on. It's time to get back to the math. This is a very propeller-head show. I'm ready. Tell me what they actually did.

**Steve:** Okay. So get a load of what the non-computer scientists who apparently...

**Leo:** Aren't they computer scientists?

**Steve:** One would wish. But wait till you hear this.

**Leo:** Oh.

**Steve:** So once again, here's what we actually got. This is the definition in the RFC of the HOTP algorithm from 2005. Once again, of course, we start with the output of the SHA-1-based HMAC. But this time, rather than viewing it as a large and, I don't know, I guess apparently intimidating 160-bit binary number, we view it as an array of 20 eight-bit bytes. The bytes of this array would be numbered 0 through 19.

**Leo:** Okay.

**Steve:** The officially standardized HOTP algorithm instructs us to take the last byte of the array, byte number 19, and mask off or ignore the upper four bits of that last eight-bit byte. Thus we'll be paying attention to only the lower four bits.

**Leo:** We're throwing out half the entropy right there.

**Steve:** Right there. These four bits will thus have a binary value ranging from 0 to 15. So we use that 0 to 15 value as an offset into the entire 20-byte array where, starting at whatever offset we have, we then take four successive bytes to get 32 bits. So, for example, if after masking off the upper four bits of the last byte and retaining only the lower four bits, we wound up with a 0, we would obtain the four-byte, 32-bit value from bytes 0, 1, 2, and 3 of the array.

**Leo:** Okay.

**Steve:** And at the other end of the range, if the last four bits had their maximum value of 15, we would obtain the four-byte, 32-bit value using bytes 15, 16, 17, and 18. Okay. So this kludge, which appears to be my word for the day, results in us having extracted 32-bits somewhere from within the first 19 bytes of the 20-byte SHA-1 hash value, where the lowest four bits determine where within those 19 bytes we grab 32 bits.

**Leo:** Okay. But those were still random bits; right?

**Steve:** That's true.

**Leo:** Yeah.

**Steve:** Okay. Now, next, believe it or not, the implementer is then instructed to set the most significant bit of those 32 bits to zero. This creates a 32-bit value which, if it were to be treated as a signed integer, would be guaranteed to be positive because signed integers use their high bit as their sign where that high bit set to "1" means that the number is negative.

**Leo:** So now it's a 31-bit number.

**Steve:** Correct. So we have...

**Leo:** One of those [crosstalk].

**Steve:** Yes, exactly. We have what is essentially a very tame 31-bit positive number ranging from between 0 and 2,147,483,648.

**Leo:** Well, it's easy to say, anyway.

**Steve:** Which fits handily into a CPU's 32-bit register. Or the integer of pretty much any high-level computer language. This makes division as simple as a single machine instruction. So the HOTP algorithm next instructs us to divide that 32-bit, guaranteed to be positive integer, by one million because the remainder of that division, when converted into a decimal number, will give us all possible six-digit numbers from 000,000 to 999,999.

**Leo:** And they're still randomly distributed in that range. Yes?

**Steve:** Ehhhh...

**Leo:** Oh, okay.

**Steve:** Watch what happens.

**Leo:** Okay.

**Steve:** So does it work? Yes. And what it sacrifices in elegance, which is to say pretty much everything, it doubtless gains in ease of proper implementation using any high-level language. I'm sure anyone could write it in BASIC and obtain the correct answer.

**Leo:** Okay. So that's important.

**Steve:** It would be, yes, it is, it absolutely, it would be very difficult to screw that up. And since interoperability among all HOTP generators, all arriving at the same correct six digits, is paramount, I guess I can see why the designers chose the kindergarten design that they did.

**Leo:** Yeah.

**Steve:** Now, you might ask "Kindergarten? Really? Isn't that being too critical?" Let's look at it. From a cryptographic standpoint the algorithm itself is really quite crappy because very little of the SHA-1 hash's entropy winds up being used.

**Leo:** Right.

**Steve:** The last byte's top four bits, as you commented, Leo, are completely ignored.

**Leo:** Out the window. Yeah.

**Steve:** And the lower four bits select just four out of the remaining 19 bytes, completely ignoring all of the other 15, which is 120 bits ignored out of the total 160. Then, adding insult to injury, of the precious 32 bits that were selected, one of those is discarded because whomever is implementing this might not know how to perform unsigned division.

**Leo:** Wow. It is kind of insulting, okay.

**Steve:** So we're going to take that off the table. So the dividend on top is forced to be positive, just to be sure. So we wind up using the entropy contained within just 31 bits of the HMAC function.

Now, by comparison, my approach of successively taking the entire 160-bit hash output, dividing it by 10 and using the remainder, takes advantage of every one, as we noted, of the available bits of the HMAC output for the determination of each successive decimal digit.

But I will be the first to concede that interoperability of implementation matters here, far more than cryptographic perfection. Dividing the extracted 31-bit value by one million to obtain a value ranging from 0 to 999,999 will absolutely provide a completely useful and highly pseudorandom result.

**Leo:** Yeah. I mean, at this point the only flaw is you over-generated entropy by using HMAC. You made too much entropy.

**Steve:** You could definitely look at it that way, yes.

**Leo:** You don't need it.

**Steve:** You generated unnecessary entropy.

**Leo:** Yeah.

**Steve:** And boy, have we just thrown it out. Okay.

**Leo:** Okay.

**Steve:** One of the features of a high-quality cryptographic hash function such as SHA-1 is that - and this is to your point, Leo - every single bit of its result has an exactly even, 50/50 chance of being a 0 or a 1. So taking any sufficiently large set of them and dividing them by one million will give us a good result.

However, if our priority, as it appears to be, is to create a super-simple, easy to implement, and highly interoperable solution, then why all the low four-bit nibble nonsense to select the set of four bytes to use? As we all know...

**Leo:** Any four would work.

**Steve:** Yes. The definition of any cryptographically strong hash function, which lies at the heart of the HMAC, is that every single one of its many bits are treated equally. They each have that algorithmically guaranteed 50/50 chance of being either a 0 or a 1. So if we're going to go the route of using a 32-bit positive integer as our dividend, it absolutely and truly doesn't matter at all which 31 bits out of the SHA-1 hash's 160 bits we select to be the dividend for our division by one million. In fact, it CANNOT matter, or we don't have a truly strong cryptographic hash function to begin with.

**Leo:** That makes sense. I mean, we do have to come up with a universal way of doing it so we all do it the same way.

**Steve:** Well, this means that an exactly equivalently strong HOTP algorithm could have told us to just take the first four bytes.

**Leo:** Take the first four bytes. You don't have to pick randomly in that big set. Just take the first four.

**Steve:** Makes no difference at all.

**Leo:** Throw out the rest. That's a good point.

**Steve:** Did this make them feel better? I mean, I worry because who were these people? You know? Either they know what they're doing or they don't.



**Leo:** Why did they do all the rigmarole of taking four bits off and then indexing into the big number and finding - it's nonsense.

**Steve:** It is nonsense.

**Leo:** Take the first four.

**Steve:** It is nonsense.

**Leo:** Oh, that's hysterical. Which makes you worry, as you should, that they were - it was a lot of hand waving.

**Steve:** Did someone's mom suggest this? I don't, I mean, not against anything, not just [crosstalk]...

**Leo:** No, Mom might be a mathematician.

**Steve:** There are a lot of cryptographically savvy moms out there.

**Leo:** Yeah.

**Steve:** But, boy.

**Leo:** That's a really interesting point.

**Steve:** It's nonsense. Okay. So...

**Leo:** It's like swinging a chicken around three times before you pick the number.

**Steve:** Oh, no. That was in the appendix.

**Leo:** Oh, my god. That's hysterical.

**Steve:** So it's a little worrisome; right?

**Leo:** Yeah.

**Steve:** Did the designers of the HOTP algorithm that we're now all standardized on not understand how hash-based HMAC functions operate? You know?

**Leo:** It doesn't matter what four bytes you select.

**Steve:** Not at all.

**Leo:** They're all random.

**Steve:** It cannot, it cannot matter.

**Leo:** They're all equally random.

**Steve:** Yes. It cannot matter.

**Leo:** Wow.

**Steve:** Okay. So the only additional observation I'll make is that it is only, now, here's the really - you thought the propeller-head was spinning fast already. It's only when the dividend on top is an exact even multiple of the divisor on the bottom that we obtain a truly evenly distributed remainder. Whoops. And the corollary to that is that the larger the dividend is than its divisor, the more evenly distributed are the values of the remainder. More than anything else, this is why I prefer my approach, because it uses the largest possible value, meaning the entire 160 bits, as the dividend.

**Leo:** And that is an even multiple.

**Steve:** Well, no, it's still not an even multiple. But, boy, is it big.

**Leo:** It's big, okay, okay.

**Steve:** The bigger it is than the divisor, the better.

**Leo:** The better.

**Steve:** Okay. So let's look at an example. Okay. We'll use a super simple example to clarify the point. Say that we want to extract a decimal digit from a four-bit source. We know that we can do that by dividing the source dividend by 10 to extract a decimal digit. So now let's look at the result we obtain from all 16 of the source's possible values: 0 divided by 10 gives us 0 with a remainder of 0; 1 divided by 10 results in 0 with a remainder of 1; 2 divided by 10 is 0 with a remainder of 2; and so on upward where 9 divided by 10 is 0 with a remainder of 9. Next, 10 divided by 10 will be 1 with a remainder of 0; 11 divided by 10 will be 1 with a remainder of 1; and so on up to 15, the maximum value that four bits can have. Dividing 15 by 10 gives us 1 with a remainder of 5.

**Leo:** And we only care about the remainders here; right?

**Steve:** Correct. But look what happened. We were asking our four-bit source to give us 10 possible output values, 0 through 9. But because four bits has 16 values, it cannot be evenly mapped into 10 results. So taking the remainder of the divisions by all possible source values, we wind up with two instances of remainders of 0, 1, 2, 3, 4, and 5; but only single instances of remainders 6, 7, 8, and 9. In other words, we do not wind up with a perfectly even distribution of all possible output values.

**Leo:** Huh. That's why you get repeats.

**Steve:** Not exactly.

**Leo:** Okay.

**Steve:** Our HOTP algorithm divides a 31-bit dividend - having 2,147,483,648 possible values - by one million. And since that total number of possible input values, 2.147 billion, is not evenly divisible by one million because it didn't end in six zeroes - it's got to end in six zeroes if it's divisible by a million - this means that not all possible six-decimal digit values produced by the industry standard HOTP algorithm will occur with exactly the same frequency.

**Leo:** Aha.

**Steve:** Now, in practical terms, am I splitting hairs? Definitely. It absolutely doesn't matter at all. It won't result in the final decimal output, which will change again in 30 seconds anyway, being usefully any more guessable. The case of generating 10 values from 16 was so horrible only because 16 was so very close to 10. By comparison, HOTP's dividend is 2.147 billion, which is much, much larger than one million. In fact, it's more than 2,147 times larger than one million.

But that said, computer science is computer science, and all of this makes for intriguing questions. If nothing else, these questions must be examined, if only to be able to judge their size and impact and to make certain that their effects will be negligible. The only thing it means is that the exact number between 0 and 999,999, the sum of them in a huge universe will occur ever so slightly less often. And it's like...

**Leo:** Who cares; right.

**Steve:** ...three or four decimal digits of percentage.

**Leo:** Okay. Yeah.

**Steve:** You know, one of them is 0.99999, and the other one is 1.00001. So just a tiny bias in the number of times, if you just took readings forever and ever and ever. But

again, for this application, absolutely makes no difference because it changes again in 30 seconds.

**Leo:** And as you pointed out, HMAC is a pseudorandom number. It's a pseudorandom hash; right? So there are probably biases built into HMAC, too.

**Steve:** No.

**Leo:** No?

**Steve:** That's the beauty of SHA-1.

**Leo:** It's really random. Oh, okay.

**Steve:** It has never shown any bias.

**Leo:** Interesting. None at all.

**Steve:** So mixing things in there, I mean, they really did the work there.

**Leo:** That's cool, okay.

**Steve:** So for me, from me, you will only ever get long division of all possible available bits of entropy, not because it necessarily matters to you, but because it's the most correct solution, which is what makes it matter to me.

**Leo:** And you're the only implementer.

**Steve:** Yes.

**Leo:** So you know you can do it correctly. And there's no onus on you to make sure that a million other people could do it correctly.

**Steve:** And I do not disagree that making a simple maximally easy implementation matters. And if that was their goal, just take the first four bytes, you suckers.

**Leo:** They didn't make it the simplest.

**Steve:** No.

**Leo:** That's the irony. They could have made it much simpler.

**Steve:** They mixed in some mumbo-jumbo that cannot have any benefit.

**Leo:** Just take the first four bytes. That's all you need.

**Steve:** It cannot, it cannot make a difference.

**Leo:** That's pretty funny. It was [crosstalk] to make this easy.

**Steve:** Maybe it made them feel better. Maybe it's spookier or something. I don't know. Whooooo. Anyway.

**Leo:** This is really worrisome. I really do wonder why they did that. That is very strange.

**Steve:** I know. It is, it is a concern.

**Leo:** Yeah.

**Steve:** Okay. So now let's return all the way back to Max's original question of any perceivable bias in the resulting numbers that might cause more identical digits than we would expect. Knowing what we know now, is that possible? No.

**Leo:** Oh.

**Steve:** It is not possible.

**Leo:** Okay.

**Steve:** Because we are - we've examined the algorithm. At its heart it takes a sufficiently large, entirely pseudorandom binary value from which we take one of 2.147 billion values and divide that number by one million. The dividend of the division, while not an even multiple of the divisor, is large enough that the divisor, than the divisor, that the remainder of that division, the number varying between 0 and 999,999, will be an extremely evenly distributed value within that range. And that in turn means that, when converted into a decimal number, the value's individual constituent digits will also be extremely evenly distributed without any possible interaction or relationship to one another.

Now, I should say, I, too, have observed the same illusion that Max and his mom have.

**Leo:** As have I.

**Steve:** And that you did before the show, Leo.

**Leo:** Yeah.

**Steve:** But I'm certain that this must be classic observational bias, where we tend to notice much less all the times when the digits do not form any sort of pattern, and tend to notice more those times when they do. But that aside, it is provable, and we just proved it, that there cannot be any non-uniform pattern. And we know that all authenticators must be using the same algorithm which we've just examined. Otherwise they would not be producing the expected result.

Now, I asked ChatGPT. I said: "Is there a term for the tendency of we lowly humans to perceive a pattern where none actually exists?" And it replied: "Yes. The general term for this is apophenia, the tendency to perceive meaningful connections or patterns between unrelated things. A more specific example of this phenomenon, limited primarily to visual or auditory stimuli like seeing faces in clouds or hearing hidden messages in music, is called pareidolia."

**Leo:** Yes. I knew that.

**Steve:** One thing, Leo, I am quite certain of, is that there is definitely a pattern to these podcasts. They routinely appear every Tuesday, come rain or shine. So everyone should expect another one next week.

**Leo:** And you might even see a face in these podcasts, if you squint your eyes a little tiny bit. That is fascinating. And now, of course, I'm looking at my authenticator, and there are 33 different codes in here. And I've already found two that are not repeating. I think it's probably about 15%. So, and you validated that it should be about 15% because it's these six digits are truly random.

**Steve:** Yeah. And...

**Leo:** Even if they're calculated in a completely absurd way.

**Steve:** And we know everybody's - yeah, oh, god.

**Leo:** I love it that - and now, you take the offs, subtract the nibble, take the offset of the lower nibble, and you go into the thing, and you get those four bytes. And you just take any four bytes, it doesn't matter.

**Steve:** Cannot matter.

**Leo:** Take the four bytes.

**Steve:** Cannot matter. And, I mean, and it worries - it's like the guy who designed his own crypto algorithm. Oh, this scrambles the bits up so good, they're never going to unscramble them. And it's like, okay.

**Leo:** It's a fundamental misunderstanding of how SHA-1 works.

**Steve:** Yes.

**Leo:** And of the generated value.

**Steve:** Yes.

**Leo:** Which is scary if somebody's writing code that uses SHA-1.

**Steve:** Yes.

**Leo:** But fortunately, they screwed it up in the right way, not the wrong way.

**Steve:** Fortunately, because it's all pseudorandom, they couldn't unscrew it up. I mean, there's no way to do something that was like really bad. It's just no better.

**Leo:** Take the hash. Take the first four bytes. You're done. Would have been a lot easier.

**Steve:** Yeah.

**Leo:** That's hysterical. But if you really care, you do it Steve's way. And you do some division and blah blah blah. Take the remainder, and you divide it some more. Take the remainder, divide it some more. You're pretty funny. Perfect Paper Passwords is a perfect example of Steve's monomania, my friends, to make sure that you are secure. Go to GRC.com. It's there still. Here we are, 17 years later, it's still there.

Copyright (c) 2014 by Steve Gibson and Leo Laporte. SOME RIGHTS RESERVED

This work is licensed for the good of the Internet Community under the Creative Commons License v2.5. See the following Web page for details:  
<http://creativecommons.org/licenses/by-nc-sa/2.5/>