Perfect Paper Passwords

Description: Steve reveals and fully describes the unique, simple, clean and super-secure one-time password solution he came up with for providing roaming authentication for GRC's employees. He also describes his own freely available software implementation of the "PPP" system, as well as several other recently created open source implementations.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-115.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-115-lq.mp3

INTRO: Netcasts you love, from people you trust. This is TWiT.

Leo Laporte: Bandwidth for Security Now! is provided by AOL Radio at AOL.com/podcasting.

This is Security Now! with Steve Gibson, Episode 115 for October 25, 2007: Perfect Paper Passwords.

It's time for Security Now! with everybody's favorite security guy, our guru, our Man About Security, Mr. Steve Gibson. And Steve, I have to ask you, with all the fires down in the Southland, I'm worried about you. Are you okay down there in Irvine?

Steve Gibson: Well, the good news, yeah, the good news is I'm really close to the water. So I can jump in if I need to.

Leo: I don't know if that was much reassurance for the folks in Malibu, but I'm glad to hear it. So the fires aren't near you? They're not in your area?

Steve: They're not near me, but I've been wondering about Elaine, who's...

Leo: Oh, that's right, she's in the grasslands there.

Steve: I think she is. I just came back from a quick run to Starbucks to refill my trusty Starbucks thermos, and I noted that our mail delivery guy was wearing a white surgical mask over his face.

Leo: Oh, it's a little smoky out there.

Steve: So he's a public servant and out in the air all day. But, I mean, it is definitely weird. It felt on Sunday sort of like the end of the world. It was so dark in the middle of the day. And in fact even today I was able to look at the sun for the first several hours of it coming up, I mean up above the horizon, it was a dark red ball because there's just so much smoke in the air. And I had heard earlier that it was half a million people. Apparently now it's more like 800 and some plus thousand people have been evacuated from their homes.

Leo: That's just awful. Just terrible.

Steve: And I guess we have - I heard another 15,000 National Guard are coming in to guard the homes and streets of areas that have been shut down because of course you don't want looting and bad guys to come back in. So, I mean, it's just phenomenal that in this day and age nature and - I guess in one case it was sparked by arson. But in many other cases they were accidental sparking events, construction workers or people with a blowtorch or something, where aided by the winds these things got out of control.

Leo: Well, our best wishes to all of our listeners in the Southern California area. And stay safe and good luck getting through this tough time.

Steve: It's so sad when you see the fires coming right up behind brand new housing complexes, nice, beautiful new homes. And it's like, oh, my goodness, they're not going to be here for long.

Leo: Nature has a funny way of reminding us who's in charge really. Once in a while she just says, oh, and by the way, you might remember me.

Steve: Before you get too cocky.

Leo: Yeah, exactly. So today we're going to - what are we going to talk about? Actually I don't know. Are we going to do the Perfect Password?

Steve: Well, yes. What we're going to do is we're going to do the long-awaited, for me, actually, second half of the podcast topic we began two weeks ago. It's been called "Roman authentication."

Leo: Roman authentication.

Steve: It's actually roaming authentication. We touched on it a little bit with the Q&A last week. And this week I'm going to explain the path I took and the solution I came up with for the need to give my employees really, really secure, I mean bulletproof security for their roaming authentication. And I've actually had the system up and running and even public, but not announced really publicly. It's really caught on. We've got a number of companies have written indicating that they're going to be using it themselves.

Leo: Really.

Steve: Two open source projects even have code published that creates a compatible implementation with mine. And in fact it's possible now to log on remotely to Macs using what I call Perfect Paper Passwords, PPP. A PAM, a Pluggable Authentication Module, has been created. So anyway, we'll tell our users, or I mean our listeners, all about this really cool solution.

Leo: So if you want to catch up a little bit, go back to 113, a couple of episodes ago, where Steve posed the problem that he was facing. It might help if you listen to that first, I would guess.

Steve: Well, and I'll give us a little quick recount of that just to bring everyone current when we start talking about this. I have a couple of little errata things. I did want to mention that - it's funny, Leo, I was listening to TWiT, or maybe it was your Windows Weekly with Paul, I'm not sure which. But you happened to mention that you had the next generation of the Sony eBook Reader.

Leo: Yes, yes.

Steve: I bought mine the second day it was out, having read that it had an improved screen.

Leo: And it does.

Steve: It really does. Actually on the show notes for this week I have a side-by-side photograph that I took of the first generation and the second generation, the Sony eBook Reader. The first one was the PRS-500. This next generation is the PRS-505. And it's got whiter whites, essentially. The blacks are as black as they were. But they managed to make the screen - improve the contrast by making the whites whiter. Although I've got to say, and you did mention this, but I wanted to - that is, on the podcast where you were referring to the new one. They say they sped up the page change?

Leo: I don't notice it, no.

Steve: Well, what they've done is, I mean, it, like, twists your eyeballs in their sockets.

Leo: I never liked the page change. You get used to it.

Steve: I guess. I'm not used to this one.

Leo: Is it worse, you think?

Steve: Oh, yeah. It's a different technology.

Leo: See, I'm at a disadvantage because I lost my 500. And so I can't do a head-to-head comparison. And I wasn't even sure that the screen was - it felt better, but I wasn't even sure it's better. But the page turn, because it's an electrostatic screen it has to refresh the whole screen - essentially, like, shake it like an Etch-A-Sketch and then redraw it. And it gives you a flicker and a flash.

Steve: Oh, they've done something wacky now. I mean, it almost looks like it sucks the centers out of the letters, leaving the outlines behind. Then they sort of twitter, and then they all upside down invert a couple - I mean, it's just like a major - like I said, it just twists your eyeballs in their sockets. It's bizarre looking. In fact, I'm considering setting up a camera and doing some page changes...

Leo: Slow it down, yeah.

Steve: No, just so I can show...

Leo: Well, I'd like to see it in slow motion to see what it's doing.

Steve: That's a very good point. I've wondered, too, because it happens quick enough that you really can't see it. But now, I mean, there's different changing technology in this one. And it's like, ooh, boy, I mean, it really gets you.

Leo: See, I didn't notice that. It felt slightly faster. But it's still annoying. You know what I do is, as I'm reading, I try to time my page turns because I know it's - if you press the button, there's a slight pause, and then it flickers, and then it goes to the next page. So I try to time it just before I get to the end of the sentence. Sometimes I miss and have to go back because I missed the last few words.

Steve: I have caught myself, on the first generation I was doing the same thing, Leo. The other thing I discovered, and this was sort of a mistake, is I notice now that the Night's Dawn trilogy is available. That's Peter F. Hamilton's monster, I mean, to call it a "tome" doesn't even do it justice.

Leo: Is that on the Sony CONNECT site?

Steve: It is.

Leo: Oh, baby.

Steve: But beware. I think it's a 19MB download.

Leo: What? It's that long?

Steve: They put all three books into one file.

Leo: Oh, dear.

Steve: And it completely chokes the Reader when you attempt, I mean, if you touch it you think - I thought it had locked up completely. It took maybe two hours for the Reader to come back online after I touched it to the Night's Dawn trilogy. It is 7,536 pages long in medium font size.

Leo: So it sounds like you don't recommend doing it this way. Once you open it, is it okay? I mean, can you go through the pages?

Steve: No. You cannot go away and come back because what it does is I think it must do a pagination of the book. And oh my god. So I'm on the first page, and I'm going to read a few pages just to sort of experience it. Then I'm going to say, okay, well, I'm not serious about reading this thing. But so this still - the reason I mention this is I want to explain to people that this has all the feel of still sort of a first generation. Even though we're at second generation, they're just not there yet. The other thing is, since I've had it for about three weeks I've been meaning to talk about, but I keep forgetting on the podcast. During two weeks of it just sitting there, it completely drained its battery.

Leo: Well, two weeks is a long time.

Steve: No. Two weeks of something not being turned on? It's not like it was on. It was off for two weeks. And when I turned it on in order to check the page count because I wanted to mention that, I immediately got this warning symbol came up and said "Battery Low." And it's like, oh my god, I mean, it was full two weeks ago, and I haven't done anything with it. So it's just - it's wrong that something is running inside this thing and burning up power. There ought to be a really off switch that makes it, I mean, literally disconnects the battery so that it can sit there for six months.

Leo: Now, on the bright side it does charge via USB port. You don't need that special charger anymore. In fact, doesn't even come with it.

Steve: It no longer comes with a charger.

Leo: That's easy.

Steve: It no longer has that standup stand that the first generation had. It is also a full USB drive, USB profile. So you're able to plug it into a system and see it as a drive. And in fact you can, for example, transfer PDFs and text files and other things to it, rather than having to route everything through Sony's equivalent of iTunes.

Leo: And I've done that on a Mac. And that's really handy because there's no Sony CONNECT for the Mac. So that's a very handy way to get stuff onto there.

Steve: And it now has an SD card slot, which is nice.

Leo: No, I think it had SD before, didn't it?

Steve: No, it was only Sony's proprietary, you know...

Leo: Not that you need more memory.

Steve: Right.

Leo: I mean, it comes with, what, 180, 192 megabytes now.

Steve: Yeah, if you were crazy enough to need - wanting to be listening to music, certainly music takes up a lot of space. And of course it does burn up your battery a lot...

Leo: Do you still listen to music? You were doing that for a while.

Steve: No, I don't listen. I did it as an experiment on the first-generation Reader just to see. And it's like, yeah, okay.

Leo: Listen to your iPod.

Steve: Exactly. And I really like the idea that when you're actually reading with the book the battery lasts for so long.

Leo: That's what I want, yeah.

Steve: And in fact it did seem to recharge way faster than usual after I got the warning sign. So maybe it actually wasn't that the battery was way down. Maybe it just needed a little hit of power or something. I don't know what was going on.

Leo: Now, fill me in on this. And I did a full review; it's going to appear on the Daily Giz Whiz. And at the time I wasn't sure. I don't remember a Jump to Page feature in the original. And now you can, in fact, go to a specific page.

Steve: They changed that. In fact, they moved the buttons from along the bottom to up along the side, which really makes sense now because the UI has the various items numbered on the side. Now you've got buttons right next to them.

Leo: Next to them, yeah.

Steve: Actually the original one had an annoying feature where the buttons along the bottom jumped you to that percentage. So if you...

Leo: Right. And you hit it by accident all the time.

Steve: Yes. I did it all the time by mistake.

Leo: And I couldn't find a way to jump back.

Steve: Well, there is, yes, you were able to use that back mode in order to go to where you were.

Leo: Oh, okay.

Steve: So "Back" did work. However, one of the things that sometimes happened also is I would leave it on by mistake. And then just because the buttons were on the face of it they would kind of get themselves pushed without intending to. So when I came back to the book it would be on and completely in some lost zone. I mean, I had no idea where I was.

Leo: But now you can jump to a specific page, too, which really, to me, helps.

Steve: Yes, you're able to dial by page, you know, 365, and it'll go to that page.

Leo: You couldn't do that before, could you?

Steve: No, there was no direct page access. You could only go by tens of percentages and then painfully step forward or backward. I guess I'm saying, you know, everything we're talking about says, whoa, this thing has a ways to go. But again, as a reader, it's a nice reader. And I did want to also mention, just while we're on the topic, the Amazon Kindle is apparently just days away from being released. The Kindle is Amazon's eBook reader. I guess it's no big stretch that a major bookseller like Amazon would want to say, hey, we want to be in the eBook business. They bought, it was Mobipocket, I think it is, which was the very first eBook source that I was using back in my earlier Palm eBook days. Maybe it's not Mobipocket. Mobisomething. Anyway, the thing about the Kindle that's interesting is that it is also a cellular modem built in. It's on the Sprint EVDO network, so way fast. And so...

Leo: Do you have to have a Sprint account, though?

Steve: My guess is that you get one as part of it. I mean, as I understand it, the way EVDO being on Sprint works is it would have to have a Sprint phone number. But you would also have an account in order to buy books through Amazon. So my guess is that part of the registration process, setting up with Amazon, would be you create an account, and then you buy the books. And I'm guessing, but I'm pretty sure that Amazon would cover the cost of downloads. So you're not seeing any Sprint usage billing. Amazon's done some deal with Sprint where, yes, you do have a Sprint phone, technically, but it's just a cell modem, and you're not going to be using it much because it's only for downloading books. But it's cool because for someone like my mom, for example, who's really not going to be docking her eBook reader onto her Mac in order to do all this, that just sort of - she wouldn't want to go through all that. The idea would be, of course, that it's a web browser, has an interface to Amazon's UI, and the entire process

of searching for books and purchasing books you do with this reader, and you need no docking, no computer. So wherever you are you could buy books, and they'd just magically appear in there, courtesy of the Sprint cell modem.

Leo: Well, that's neat.

Steve: Yeah.

Leo: Yeah, I guess I'll get one of them, too. Although I have to say the 505 is now close enough that I feel pretty comfortable with it. I'm reading a lot with it. I've been reading a lot for the last - got it three or four days ago. I've been very happy with it. I mean, it's not quite - I want paper white with black text. But it's close.

Steve: It actually is enough better that I'm no longer finding myself annoyed.

Leo: I wish the scaling were infinitely variable, too. Because some books you only have small and medium. It's hard for me to read the small or even sometimes the medium text. I like it the large text.

Steve: Yeah, but large is really large.

Leo: It's large. And I wish there were kind of a way to kind of slide, have a slider that I could go...

Steve: I agree. I think that...

Leo: Because it's a PDF. It should be able to do that.

Steve: Well, it's got a clearly rescaleable technology built into it.

Leo: Right.

Steve: So you're right, it definitely could do that. And that would be nice.

Leo: And for people with really low vision you could get even bigger, and that would be nice. That's the real benefit of this. I'm too embarrassed to buy large-text books, large-type books. But this way it can grow with my failing eyesight, so to speak. Anyway, enough of the eBook update. But I think, you know, we're interested in it, and I hope others are, as well.

Steve: We've got a lot of good feedback in all of the times that we've talked about eBooks. I know that there's certainly a cross-section of our listeners who are interested in this evolving technology.

Leo: If you listened to TWiT you probably heard Jerry Pournelle talk about the fact that, to him, the iPhone was very close to being the eBook reader. And that's an interesting point of view. It really is a very good screen.

Steve: Well, and his point is that there you're repurposing something that you already own.

Leo: Right.

Steve: That's what Jerry was talking about was that, hey, I've already got an iPhone in my pocket. So wouldn't it be nice if I'm stuck in line or I'm in an airport or something that I could read a book that I'm wanting to read anyway on the device I already have.

Leo: And I have to say the text is much more legible on it. I can read a much smaller typeface on it very easily. So...

Steve: Well, and contrast has a lot to do - I don't know if you've messed with that. But contrast has a huge amount to do with the size the text has to be. As contrast is reduced, you really need a larger font in order to make up for that.

Leo: Ah, that explains it. Yeah, because I don't normally - I'm not normally that blind. But I cannot - the small type, I just cannot read it. On the other hand, I'm reading smaller type on my iPhone, no problem at all.

Steve: And I've noticed that if I've got the Sony Reader under bright light, I can read on the small font...

Leo: Oh, yeah, it's much better, yeah.

Steve: ...with no trouble. But not if it's dimmer, so.

Leo: I suspect what you're going to see is, you know, Apple's announced the availability of third-party applications soon. And, well, starting in February they're going to tell people how to do it anyway. And so I imagine by June of next year you'll be able to buy them. And I bet you that'd be one of the first things you'd see is an eBook reader. Just makes sense from Apple's point of view. They could add it to the store, be an easy thing to do.

Steve: And as I understand it, anything that the iPhone gets, the iPod Touch gets.

Leo: Yes, good point.

Steve: That's my sweet spot because I really - I think the iPod Touch is nice, if I could only get my WPA key into that darn thing. I think what I'm going to do, we actually had a lot of readers, I mean a lot of listeners, who suggested various solutions. Some have actually managed to enter a GRC-style Wi-Fi Perfect Passwords from the GRC.com/passwords page...

Leo: How?

Steve: ...into their iPhone or their iPod Touch. They get the thing, copy it to a text file, break it into four-character-long sequences, so they can kind of do them four at a time...

Leo: That's ridiculous. I'm sorry.

Steve: And they said, yes, it took me seven tries, or yes, it took me 20 minutes.

Leo: Forget it.

Steve: But I was able to get it typed in correctly. And I'm just thinking, okay, that's wrong. I want copy and paste.

Leo: Copy and paste. Although I have to say I was playing with the Blackberry because I was thinking, after you said that, I was thinking about that. And the Blackberry not only does have copy and paste, but it doesn't hide the key as you're entering it. Because the new Blackberry has Wi-Fi built in. And it's much easier to type it with that keyboard. You could probably do a pretty good job of it with the Blackberry, even if you didn't have cut and paste, but of course you do.

Steve: Well, you know, I'm a Treo user. So I'm a fan of a physical keyboard with actual buttons on it. I just, you know, everyone says, eh, well, you know, the iPod Touch works, and it survives without needing a physical keyboard. But it's like, eh, I wish it had Bluetooth. Then I could Bluetooth to a real keyboard, and then I could type it in with no trouble at all.

Leo: Oh, wouldn't that be clever.

Steve: But no.

Leo: Well, we'll see an ecosystem around it. I mean, it's brand new product. And I think that just as there's an ecosystem around the iPod, there'll be an ecosystem around the Touch. And the iPhone. Do you have any addenda, errata, corrections you'd like to pass along?

Steve: I don't, although in my eternal quest for interesting SpinRite stories I do have one. I'm not sure what to call this one. We'll have to consider maybe cutting this out of the podcast because I would want to call this one, "SpinRite Got Me Laid."

Leo: Oh, my goodness. Let's see, "...Made My Love Life Better." How about that?

Steve: Okay, it definitely did. This is from Mike.

Leo: if this is true, this is the best endorsement you've ever had.

Steve: This is Mike in Tucson, Arizona. Actually he wrote today. He said, "I wanted to share this SpinRite success story with you."

Leo: Oh, man.

Steve: This gives a new meaning to the word "success." He said, "I met a girl online, and we chatted many times before finally meeting. Just before we met, her son's laptop would no longer boot, and she asked if I could help. So I purchased a copy of SpinRite...." So he purchased a copy of SpinRite for his first date. And he says...

Leo: Now, that's a nerd.

Steve: "...for our first date the next night. Even...." Now, get this. He says, "Even though I didn't think the date went that well, she trusted me enough to let me take the laptop home to work on. It was dead anyway. Later that night I ran SpinRite. And sure enough, it fixed the problems, allowing the laptop to boot without trouble. So the next day I proudly took the fixed laptop to meet her. She was so...."

Leo: Let's leave it at that.

Steve: "She was so happy and thrilled that, well, two months later we're still dating and having a wonderful time. Thank you, SpinRite, and thank you, Steve."

Leo: It's good for your love life, ladies and gentlemen. Wow, is that a ringing endorsement. That's hysterical.

Steve: So thank you, Mike. I really appreciate you sharing the story.

Leo: So let's take a look at the Part 2 of - you know why I call it "Roman authentication"? Because there's a device in the gym called a "Roman chair." Have you ever used the Roman chair, Steve?

Steve: I haven't.

Leo: It is the most - it's clearly a device invented by the Romans, probably to torture Christians. I don't know. You get your legs in there, and you hinge over at the waist, down to the ground and up again. It's like backwards sit-ups. And just for some reason Roman authentication reminds me of my Roman chair. It's actually a fun exercise. Now, you're talking about roaming authentication. And again, if you heard 113, you know this was Steve trying to solve a problem with getting his employees online when they were out of the office. Why don't you recap what we talked about?

Steve: Right. The idea, well, where we left off - and the goal of this is to allow - and it's the kind of thing we've talked about a lot before. We've talked about multifactor authentication. I wanted a means that would allow Sue or Greg when they were not at their home base location, that is to say, with a laptop, to be able to access pages at GRC on the GRC server which are highly privileged. Basically the back end, for example, of our eCommerce system, which would allow Sue to issue refunds, to find people's replaced or lost SpinRite serial numbers and purchase information in order to email people the ability to redownload SpinRite if they're away from home, basically to do all the things that she's able to do when she's authenticated just by virtue of being at her home IP, which is authorized, but from a different IP.

So I did this in a couple stages. The idea is that, when she's at home, and she accesses GRC from her laptop, it notices that she's at her residential IP. And if she isn't already carrying, that is, if that laptop isn't already carrying an authorization cookie, it says, hey, do you want to authorize this machine for roaming authentication when you're not at your home IP? So if a friend came over and for some strange reason she accessed GRC's privileged pages from a friend's machine, it would pop that up, and she would say, no, I don't want to give this roaming authentication. I mean, that seems unlikely, but it's built in. So that question would come up. And on her laptop she would say, yes, she does. And in fact she would even say that on her desktop machine because from time to time the IP might change on her cable modem that is provided by DHCP on the cable modem, in which case she would need to be able to still access with a different IP because then it would think, with her IP changing, that she was no longer at home.

Now when she's out at Starbucks, for example, roaming, and she attempts to access these privileged pages at GRC, the server will see that a request is coming in for a privileged page. It'll notice that with that request is this permanent authorizing cookie, which will only be sent over an SSL secure encrypted connection. So that's protected from anyone snooping on the data track. If she were accessing GRC's nonsecure pages, that cookie would not be offered by the browser. So in seeing her do this, she is prompted through some mechanism to prove who she is because in a multifactor authentication mode we want, as we've talked about before, we want there to be several different things which are being used together to prove the identity of someone. So the fact that she's using a laptop carrying this secured cookie, which could only have been issued when she was at her known authenticated IP, that demonstrates that this laptop was once there. So this is something she has, that is, her laptop, that is allowing one stage of this authentication. She needs to enter also a secret password, that is, a nonchanging secret password to say this is - not only is this request coming from the laptop, but it's also coming from me, somebody who knows my own secret passphrase.

Now, the problem with relying on only those two things is it would be subject to some sort of recording attack, that is, a replay attack where a keystroke monitor were monitoring her entering in this passphrase, that being something that isn't changing every time. So that's where we left off with Episode 113. I wanted to come up with some means, some way, another factor of multifactor authentication which would prevent even somebody with perfect information about the log-on process, that is, somebody with a camera, somebody recording keystrokes, somebody recording mouse clicks, I mean, in a mode where there's perfect knowledge of what I and my employees do to log in.

Leo: Even a keystroke logger would be thwarted by this.

Steve: Yes. Perfect knowledge, full knowledge of what they're doing to log in. And that party still could not duplicate what they do. So my first thought was, okay, some sort of a puzzle. And I did talk about this in Episode 113 two weeks ago, the idea being that the server would challenge them with something on the screen, like jumbling up numbers and letters, and only a GRC employee would have the magic incantation, like know for example that you sum the digits until you get to a single one, you reverse, and then you interpose that in between the alphanumerics, and you sort the alphabetic in reverse XYZ order or something.

And so I thought about it for a couple days, trying to come up with something where it would be knowledge which would not be obvious. And the problem is that I want - I don't want to have any compromise in security. So I wanted to come up with something where somebody looking at the answer and looking at the challenge would not be able to reverse engineer it. And I couldn't. I just, you know, I looked at it and I thought, you know, if I say, oh, this is really hard, no one's going to get it, it's like I'm cheating. And I can't allow that in this case.

So I thought, okay. We need essentially something that they have. We want to prove that they have something separate from the computer because, again, laptops are being stolen all the time. So it's got to be separate from the computer. Now, we've talked about, of course, the PayPal, VeriSign, RSA-style crypto dongles. Except that all of those require that you use a back-end service. And I didn't want to be going to VeriSign or any third party for verification. Not that I couldn't, but first of all, VeriSign hasn't ever followed through on their suggestion that they'd be willing to show me how to do that or give me access to their back end. So it's like, okay, well, I don't have access to that, and I didn't want to depend upon it. So it's like, what could they carry that would comprehensively prove that they, and they alone, have this.

So I started with this idea that we had glanced on, we'd sort of heard of other companies doing this same sort of thing. And I just said, okay, let me think about this more because I hadn't thought about it. And that's the idea of some sort of a printed sheet, some sort of a - something printed.

Leo: So they carry it with them.

Steve: They would carry it with them. It would be in Sue's purse or in hers or Greg's wallet, or mine in my case. And in fact next time I see you in Vancouver, Leo, I'm going to show this to you because it's all done and working. And so the idea is something that is carried with you. So I thought, okay, let's - and we know how pseudorandomness works. We know, for example, that you could have a cryptographic key, and you could put in a counter into a cipher that's driven by the key, and out will come random-looking but deterministic data. And, for example, that's how the new Perfect Passwords page at GRC works. You may remember that I was using a weak random number generator before, not knowing that it was weak. It was from RSA. I would have thought that it would have been good. But it turns out an analysis of lots of its output demonstrated its entropy was not as high as it could be. So I redesigned one myself using the Rijndael cipher, basically using a counter as input to it. And what comes out is fantastically random. So it solved that problem. So if we have a key, and we just run a counter into a cipher, a good cipher, what comes out the other end is really high-entropy random junk.

So first I thought, okay, so say that we take this random junk, and we just print it on an 8.5 by 11 page. So we have this huge grid of junk. And so I thought, okay, well, it would be easy to prove that I had that page because every one of them would be different. And so, for example, I thought, okay, say that I just picked a random character in this massive grid of gunk and typed in five or six of them in a certain direction. I could even go in any direction I wanted. I could go up, down, left, right, diagonally up to the left, up to the right, down to the left, down to the right, sort of in any direction. And so I just choose a random spot and type in a series of characters that are on the page.

Well, then the authentication mechanism would be that it would have - the server would know the same key that was used to generate the page originally. It would in memory create a facsimile of that page. And then it would look at the first character I typed, scan this page to find an instance of that first character, and then look at the second one and see if there's a second character in any of the eight locations around that first one. That would give it a starting direction. And then it would march in that direction to see if it was able to match all the characters I had typed. And if it failed, it would then continue scanning for another instance of that first character and perform the same search until it determined whether I had typed an X

number character-long string that appeared on that page. And it's like, okay, well, that would work. How's that?

Well, the problem is that we would not know for sure that we had never entered that particular sequence before. That is, humans tend to be repetitive. Something would have drawn our eye to a certain spot, a certain character we like, a certain reason for going in a certain direction. And there's a chance that at some point in the future, unintentionally, we would choose the same string. Or, if not the same string, maybe even we would hit on the same substring. So, for example, if we were typing in six-character chunks from this blizzard of randomness, if we typed in five of them by starting one over and then typed six, well, then, again the challenge is perfect information. Somebody who is able to acquire everything we do cannot gain anything from that.

Well, if we happen to repeat five out of the six characters, just by starting one character off, then that would mean that there would only be a very small set of tests that needed to be done, basically use all those five and test every sixth character, making it very vulnerable to a simple brute force attack. So I said, okay, that won't work. We can't just use - we can't basically allow the user to choose where they start. So then I thought, okay, well, how about if we have like a coordinate system, and the server says start in a certain location and go in a certain direction. And then next time you're prompted it gives you a different coordinate. Well, it's like, okay, that would work. Except that again we have this problem of reuse.

Essentially I came to the conclusion pretty quickly that there is no safe way to ever allow re-use of any of the data on the page, even sort of like substring re-use. I couldn't even start, deliberately start a couple characters over and have any of the characters I'd typed in before overlap the characters I was typing in now.

Leo: Because that would weaken it. It would make it easier to guess.

Steve: Exactly, because it would fundamentally weaken the whole notion of proving that I'm the one who has this because it would open it to various sorts of brute-force attacks. So I settled into, okay, we need a system where the user will under no circumstances ever be asked to enter the same string a second time. And I ought to mention that, first of all, just so people don't get confused, I'm fully aware that I haven't invented one-time password systems. FreeBSD has something called OPIE, which generates...

Leo: Opie?

Steve: Opie, exactly. I think it stands for One-time Passwords something Everywhere, OPIE [One-time Passwords in Everything]. And it works by drawing six words from a - six typically four-character, but sometimes fewer, words from a dictionary of 2048. And you string those together. Oh, and also I've even seen some commercial implementations of systems that I've just rejected for the reasons I rejected them. I mean, they're...

Leo: Well, OPIE reuses content, so there's one reason not to use OPIE; right?

Steve: That's a problem. Although one would hope that it doesn't reuse the same set of words. But there are some commercial one-time password systems where the design of their grid is such that you could be led into typing in some characters that were a substring of what you had typed in before. And that's just not acceptable.

So then I had this notion, okay, if the rule is - and it has to be - we can never ask the user to enter anything they ever did before, then what we wind up with is a sheet of, essentially, columns of characters by rows of characters where you have, for example, columns A, B, C, D, E, F, G, all the way through whatever, and then rows. And in fact the first one I designed was - it was huge. It was 8.5 by 11. And it was A through P. And it had 50 lines. So let's see [counting 1 through 15], so it's 16 rows by 50 lines of substrings. So a ton of them. So it's like, okay, well, we're getting there. So then it was, okay, how do I design these strings? Well, I like the idea very much of their being as short as possible. That is, I wanted not to have to type something that looks like one of those Windows XP license keys, you know those insane things that are five groups of five with letters and characters, because that's just a pain. And there's a high chance of getting typos. In fact, I have an older friend of mine, this might as well have been DRM because it was just impossible for him to type these security keys in every time because they're just too long, and there's too much opportunity for making a mistake. So I wanted them very short.

Well, having them short and not having them be insecure meant that I needed a larger character set. That is to say, for example, all these tokens we've been talking about, they're all six-digit tokens, meaning that they can go from 000000 to 999999. In other words, there's exactly one million possible combinations that the PayPal token, the VeriSign token, the RSA token, they all use six digits. And so each character can have 10 different combinations. It's 6 to the 10 - I mean, sorry, 10 to the 6th, 10 to the 6th power, or a million possible combinations. Obviously, 000000 to 999999.

So what I decided was I wanted a larger alphabet. That is, I wanted more possible characters in every position. And we need pretty much, for practical purposes, or certainly ease of implementation, it's nicer if you have a power of two. That is to say, for example, if we used hex characters, then we would have four bits per character, and we'd be A through F and 0 through 9, which is the hexadecimal character set. I wanted more than that. 32 characters would give us five bits, five binary bits of value per character. But 64 gives us six bits.

So what I designed was a 64-character alphabet that is case sensitive. So, for example, we've got uppercase A through Z that gives us 26 characters. Lowercase A through Z gives us another 26. So we have 52. And then we only need an additional 12 in order - no, wait. 52, yeah, an additional 12 to get us to 64. Except that the problem with that is in many typefaces there are - it's easy to confuse, for example, lowercase L with the numeral 1, or uppercase O, alphabetic O, with the numeral 0. So then I went through and I eliminated anything that was visually confusing. So, for example, there is no uppercase O or 0, even though I could have allowed one of them and have the system be smart enough to know what the user meant. Just the fact that there were confusing possibilities, that would create some anxiety on the part of the user of this system. It's like, well, wait a minute, is this an O or a 0?

Leo: Yeah, I think that's very nice because same thing with 1 and lowercase L; right? I mean, it's always bugged me.

Steve: Yes. And there's a whole bunch of them, it turns out. And then, as I eliminated those, the alphabetic portion got shorter. But that meant I needed other special characters. So, that is, I couldn't just use numeric, you know, I already eliminated 0 and 1. So then I had 2, 3, 4, 5, 6, 7, 8, 9. Well, those are very distinct. But I needed - now I think I needed another 16 or 17 characters. So I used exclamation point, at sign, pound sign. I decided to stay away from dollar sign just because it sort of seemed, well, I wasn't sure that every keyboard would have a dollar sign because in an international keyboard there might be some other symbol. So I carefully chose an alphabet of 64 characters such that they're very clear visually. So when you see them there's no confusion.

Now, since each character in this scheme gives you six bits, you don't need many of them. In fact, these cool things are only four characters long. So there, for example, uppercase X,

uppercase M, uppercase E, and then the numeral 8. Just four characters long. Now, the first reaction might be, wait a minute, that's not enough. It's not enough characters. Well, each one gives us six bits of value. So four of them gives us 24 bits. Well, 2 to the 24...

Leo: How come they don't, just out of curiosity, give you eight bits of value?

Steve: Oh, because in order to get eight bits of value we would have to have a character set of 256 different characters.

Leo: Oh, I see. So that's because you've eliminated some of those characters you only get six bits.

Steve: Well, because ASCII, you know, printable, there aren't that many really distinctive printable characters.

Leo: I get it. I get it.

Steve: Yeah. So each character gives us six bits. But if we use four of them, only four gives us 24 bits of data. Well, 24 bits is 2 to the 24, which is 16.77 million possibilities.

Leo: Right, right.

Steve: So consider that the PayPal, VeriSign, RSA tokens, those are only a million possibilities.

Leo: Because it's just digits. So by giving it some of these characters, you've really improved it quite a bit.

Steve: 16.7 times more secure, Leo.

Leo: Yeah, yeah, yeah.

Steve: So any given token, for example, a value coming up on the RSA or PayPal token, there's a one in a million chance of randomly guessing what it's going to be showing. In this scheme there's only a one, less than one in 16.77 million, almost 17 million, chance of randomly guessing. And what's actually sort of fun about this is they really look like little crypto things.

Leo: Yeah. Well, it's easy to remember, frankly. The human brain can do those four letters or combination of four letters and numbers much easier.

Steve: Well, actually they are not easy to remember because they're not words.

Leo: No, but that's easier than 10 digits.

Steve: Oh. What it's easy to do, it's easy to transcribe it. And that was the whole point is you wanted to be able to look at it and type, look at your card where this was printed and then enter it into the screen in order to prove that you are who you say you are. So the way the system works - and this is on GRC.com, all up and running sort of in a demo mode at GRC.com/ppp. You can just put in GRC.com/ppp. The system will force you to be over an SSL connection. So it'll bounce you to, starting with HTTPS, so https://www.grc.com/ppp.htm. That's the formal URL. But if you're just, you know, just put in GRC.com/ppp and it'll take you to the same page. You can then enter in a passphrase which is only here for the purpose of a demo because a passphrase is not secure. That is, the only possible attack, the only known possible attack on this system is a weak passphrase. So no actual implementation uses passphrases.

Leo: Because they just generate a random 64-bit or 64-character string, or...

Steve: Ah, no, even better. The actual key that drives this, I took the strongest version of Rijndael, the AES encryption. The strongest version uses a 256-bit key. I then added an additional 128 bits, which is added to the value of a counter. So you take the value of a counter, which always starts at zero. You add a 128-bit value to that, which is part of the master key. And then the other 256 bits is the key for the Rijndael cipher. So the 128 bits from the counter, added to the 128 bits from what I call the "sequence key," that goes into the Rijndael cipher, and out comes a random, that is to say a pseudorandom, very high-quality pseudorandom 128 bits. That is then taken in chunks of 24 bits. So each chunk of 24 bits is then converted into one of these four-character, what I call a "passcode," and then printed out on what I call a "passcard," which is one of these cards. So because 24 does not divide evenly into 128, you end up getting five and a third passcodes for every 128-bit cipher. So you generate the first five. Then you take the last byte of that one, you increment the counter, and generate another 128 bits that allows you to - and you take the first two bytes of that one in order to generate the sixth passcode.

Anyway, all the math is done. All of it is documented on the site in a series of pages that explains how this works. And I have pictures and diagrams of the whole crypto system which are good enough that two implementations have been written from my documentation to implement this. Now, from day one I produced all the code. There's a DLL, a Windows format DLL. It is - I think it's 16K that is the entire crypto system for this Perfect Paper Passwords system. It's got a series of functions which you're able to call, that is, anyone who wanted to implement this for themselves could simply call the Windows DLL, and it does all the crypto for you. And basically you give it a buffer of empty space, and you say give me X number of passcodes starting with this one, and the crypto system does that. And you also give it this, what I call the "sequence key." Remember that I was saying that we had 128 bits plus 256 bits. So the whole thing is driven by a 384-bit master key. Basically there's that many. I think it's something like 10 to the 38. I mean, it's a phenomenal number of possible sequences because of the way I designed this. So, for example, if you were to create an account somewhere where you wanted to authenticate yourself, that authentication server would randomly generate a secret 384-bit sequence key. And that sets the sequence for all time of these passcodes which are generated. And then you'd say, okay, I want to carry - oh, I forgot to mention that at one point in this, remember I had an 8.5 by 11 sheet of these.

Leo: Right.

Steve: Well, I realized that in any normal use that would probably last you about three years.

You know, it's ridiculous.

Leo: How many was it?

Steve: Let's see, it was 15, I think I had 16 - let's see [counting].

Leo: Like 32 or something?

Steve: Yeah, 16 by 50. So...

Leo: 800. That's plenty.

Steve: Yes, exactly. And then I thought, okay, well, how many are you going to use per day, and how many days do you want this? So anyway, what I realized was convenience trumped longevity. And so I came up with a design for a credit card-size card, which is what I call the "passcard." And it prints out at exactly credit card size, and three of them. On the GRC site, anyone who's curious can go there, you put in a passphrase, you tell it which card number you want to start with, and you start with number one. And so, for example, the system will give you that card and the next two. It prints them close to each other, so you could either cut them out individually, if you only wanted to carry one around in your wallet; or what I'm doing is I realized it would be cool if you just cut out all three together and then you fold the two in so that basically it's back to a credit card size, but you've got essentially three of what I call these little passcards.

Leo: Is this basically a one-time pad?

Steve: Well, it certainly - yeah. I would say it's a one-time pad. Now...

Leo: Because you're only going to use this code once, and then you cross it out.

Steve: And that's the key, yes. The server - so when you create an account it resets your passcode number to zero or to one, the first one. Actually the passcodes are zero-based, so it resets to zero. And it comes up with a random sequence key that is secret. You then say, okay, give me the first three cards, or however many, depending upon what the implementation wants to do. You could just ask for one at a time or whatever. So it generates the first 70 passcodes. In this redesigned smaller card I've only got columns A through G, which is seven columns, by 10 lines, which is 70 codes. Even then that's going to last you a long time. Every time you authenticate, for example, come up to a screen that says, okay, prove that you're Steve. I've already told it my username and my passphrase to tell it that I know something that means that somebody couldn't discover the card in my wallet and, like, Xerox it, for example. You still need to have multiple factors of authentication, even with a system like this, because you need to guard against inadvertent disclosure of this information. Now, remember that Bruce Schneier really believes that it makes much more sense to carry around something like this than it does to try to memorize a weak password.

Leo: Right, right.

Steve: He says users are already very good at protecting...

Leo: Physical things.

Steve: ...physical things, bits of things written down on paper.

Leo: Credit cards and, yeah, right.

Steve: Exactly. And so my idea was you would just slip this in your wallet behind a credit card, and that's where it would be. And you're generally pretty good about keeping track of that kind of thing. But you want to make sure that someone who knows you are doing this couldn't sneak up and get to your wallet when it was hanging in your coat pocket on a coat hanger and, for example, copy some of these down or Xerox it or something. So you still need - you need something other than this that only you know, which is secret, so that this gives you protection against that being lost.

It'd be similar to someone, for example, pushing your PayPal token, reading out the numbers, and then running over to their office and logging in as you. You still want your log-in information to remain private. But so the idea is the server knows where you are in the sequence. If you use, like, the first five passcodes, then it knows you're on number six. And so in my implementation for GRC it says, "Please type in passcode 1E on card 1." And it's because there's no reduction in security in it asking you for the code it wants. That is, you might have a system where you're supposed to just keep track of where it is. But I think it makes more sense to have the server prompt you for the one it wants. And since never ever ever will it ever ask you for that again, in my implementation I'm just crossing them off. If I have a pen with me I'll just scratch it out because it makes it that much easier for me to see the next one. And if somebody were hacking my log-in and were guessing passcodes, then there might be some possibility that I would want to know if I was being prompted for one further along the sequence than I expected. So that would also give you some indication of somebody potentially doing something nefarious behind your back.

One of the other cool things is, since the system runs forward forever, and you're never being asked about any prior codes, not only can you scratch them out, but if you think that a card may have been compromised, you can simply tell the system, advance me to the next card. I want to obsolete this card. Or say it's getting kind of ratty or tattered, or you use it so infrequently that it's fading out, for example, or you printed it on an inkjet printer and the ink is bleeding, or it got wet or something and the ink bled, there's nothing wrong with saying just cancel these cards, print out from cards I've never seen before forward, in which case you're starting further on in this virtually limitless forward-moving sequence of passcodes. And in moving this pointer forward you've obsoleted all the passcodes that have come before, and then you're standing there moving forward into the future.

So essentially it's a clean and elegant one-time password, or one-time pad system, as you mentioned, Leo. It uses very short tokens, which are cool looking because it uses them from a large alphabet that looks sort of spy-like, you know, capital G, lowercase G, 2, lowercase V. I'm actually reading from some that are onscreen right here. And anyway, I encourage our listeners to go take a look at it, at GRC.com/ppp. Several people in our newsgroups thought it was a cool idea. There is an open source implementation written in C. My code is all in Assembler. The DLL that implements the whole crypto system is 17K. And then...

Leo: Wow. For you that's huge.

Steve: Well, actually 4K of that is an Authenticode signature because I wanted to Authenticode sign it.

Leo: [Indiscernible].

Steve: Yes, but that's the full Rijndael cipher. It's written in Assembler. It's an Assembler implementation of the SHA-384 hash because I used the SHA-384 both for using these passphrases to generate the 384-bit master sequence key, but also the system is able to generate master sequence keys by using a whole bunch of machine-specific and highly volatile information, like the number of clock cycles the processor has had since it was last reset, since its last hardware reset, in order to allow it to be used. Essentially, this DLL could be the heart of anyone's implementation of a secure one-time passcode PPP crypto system, or authentication system. And one of our newsgroup members has written a full implementation which is completely compatible with sort of the reference implementation. That is to say it generates, given the same 384-bit master sequence key, it generates exactly the same passcode sequence as mine does.

Leo: Fascinating. I love it that people have taken this up and started doing their own.

Steve: Well, in fact we've heard from a bunch of companies who have said, hey, this is really great. We're going to protect our database with this system.

Leo: Wow, that's great.

Steve: I mean, it really is secure. It can handle any number of users. The passcodes go forever. And if somebody at some point got tired of printing out, oh, I'm up to Card No. 492, you could also reset yourself at any time. You would never go back to the same master key and the beginning

passcode. But since you've got 384 bits' worth of them, you just do another random master sequence key and reset the zero so you restart at the first card any time that you want to. So it just ended up being, I mean, it's what we're using. A bunch of other companies are using it. As I mentioned, there's an implementation as a Pluggable Authentication Module that allows people right now, there's a guy who did it, is using it to log in remotely to his Mac using this Perfect Paper Password system for...

Leo: Really. How does he get the Mac to know what the password should be?

Steve: I haven't looked at his implementation, but he...

Leo: He must have written a server version of it, too; yes?

Steve: Well, there is something called this PAM. This Pluggable Authentication Module is a standard, and the Mac OS X supports the PAM standard.

Leo: Oh, okay.

Steve: So he's done that, and he's plugged it into his Mac OS X in order to allow him to use the system for secure log-on.

Leo: See, many of us are going to start wanting to use this because Leopard, which comes out tomorrow, has built into it this get Back to My Mac feature where you use .Mac to log in to any Mac that's online. And so this will be very important to have a secure log-in through this system. So I'm not sure how secure their built-in system is, but this might be a nice thing to look at adding on top of it using PAM.

Steve: Well, again...

Leo: So PAM would know automatically that you - how would you modify the PAM to know what your passwords are going to be?

Steve: I can't answer the questions, Leo. I don't know anything about PAM or, I mean, I know about this system which he implemented, but I don't know about...

Leo: So somebody's written a plug-in, and that's what you have to find.

Steve: Well, no, I mean, we're linking to it. And so, yeah...

Leo: So just go to GRC.com.

Steve: Yes. On the PPP pages at GRC.com we've got the full documentation of this, a sample that allows people to print cards, see what they look like. And I've been trying to say also that not only do I have the DLL, I have an EXE. The EXE is 11K. What it does is it just exercises the DLL. So it's a command line version of the entire PPP system. Because, for example, people who wanted to implement this for, like, securing authentication to their own websites, if they had PHP or Perl and didn't want to be calling into the DLL, it's very easy in any of these languages to shell out to, essentially to an executable. So you're able to pass the parameters to the EXE, and it will do all of the Perfect Paper Password crypto and give you back the results.

Leo: Perfect.

Steve: So, yeah, it's just - again, I don't want people to be saying, oh, Gibson thinks he's invented something really amazing, and this is just one-time passwords, been there, done that. It's like, okay, yes, except that there are some cool aspects of this. I just, basically I did my own GRC-style implementation because I wanted to. And it uses strong crypto. It's got a lot of neat features that makes it very comfortable and easy to use. It's what we're using to provide us with secure roaming authentication, and it will be part of the authentication system in a forthcoming GRC product, as well.

Leo: Oh, okay.

Steve: You betcha.

Leo: And unlike a one-time pad, for a one-time pad to work you have to make two copies, and there has to be a physical transmission of one of the pads. This doesn't require that. It's algorithmic.

Steve: This, exactly, it uses very strong pseudorandom sequence generation driven by a really long key, a 384-bit key. So that cannot be brute-forced. Earlier on I mentioned that I am formally discouraging the use of a passphrase to generate the key. I'm only doing that at GRC to allow people to...

Leo: To play with it, yeah.

Steve: Exactly, to play with it. Because the only attack that would be known on this system would be to use a weak key, that is, a weak passphrase, and do brute-forcing of a passphrase to hash that through the SHA-384 to generate a key and then try to find somebody's passcode sequence by brute-forcing it. I mean, even that would be a lot of work. But that's the only weakness, which is why any real implementation will simply generate a 384-bit key at random, and then the only way it surfaces is through this sequence of passcodes. And there's just no way to reverse engineer that back to the key.

Leo: So there is still a handoff, though. There has to be something that is mutually understood, which is this initial key; yes?

Steve: No. The user never gets the key. The only thing they get is...

Leo: Is the paper.

Steve: Exactly. When they've authenticated themselves to the server and proven that they are who they say they are, then they can request that these passcode pages - well, yeah, passcode pages...

Leo: Be generated, yeah.

Steve: ...are presented on their web browser over an SSL connection. And then they print them, and then that's what they carry with them subsequently.

Leo: So that's what - and SSL is nice because there is that transmission, but that transmission is done over a known secure channel.

Steve: Right, to a known computer. For example...

Leo: Oh, that's right, they have to be authenticated first, yeah.

Steve: Yes, they've got to be authenticated. And I do not, again, for the sake of heightened security in my own implementation, I do not allow a roaming laptop to receive those passcards.

Leo: Ah. You've got to print it locally first.

Steve: You have to print it locally. And that's one of the other nice things about crossing them out is that it's very easy to see when you're nearing the end of your passcard, to remind yourself, oops, I'd better print some more. But again, you're always able to print them. And that's one of the things I like about this system is that the approach is it's a web-friendly solution. No one's having to mail you these passcards in advance. Any time you want more, you say I want more. And it says, okay, here's your next...

Leo: So now that you've - okay. So if you're using the passcard, you can get it from a laptop now.

Steve: Correct. I'm now - Sue, Greg, and I...

Leo: So you've now been authenticated to such a degree that you trust the laptop to print another set of cards.

Steve: Oh, no, no.

Leo: You still have to go home to get the cards.

Steve: Again, in my situation, because I do have this notion...

Leo: They come home all the time.

Steve: I do have this notion of a home IP versus a roaming IP. In my particular implementation I just decided, let's just not allow them to see passcards when they're on the road. You've got to be home. And in fact that's one reason it's nice to sort of print all three of these in a big chain is then you've got 30 rows by seven columns. You've got 210 passcodes. They're going to last for a year. So I don't think I see anybody using these things up very fast.

Leo: I'm very interested in - we're going to do a question-and-answer session next week, but maybe the week following we can talk about using this, or if you needed to use this, for Back to My Mac, which is something that I think is going to be a very cool feature of Leopard - we haven't seen it yet, we'll see it tomorrow - which allows you to, as long as your Macs are Leopard-based, to find any of your Macs that are connected to .Mac and get to them, display them. It's remote access, basically, built into the Mac. It uses - it says in quote - and this is the only problem is it says it uses password-protection and advanced data encryption, but it doesn't give much detail on that. I'd be very curious.

Steve: My guess is, Leo, it will not use one-time passwords. And that's absolutely what you want.

Leo: Right.

Steve: And so if this allows you to - if Back to My Mac can accept a PAM plug-in, then I wouldn't be at all surprised if we can make this work. And if not, maybe we'll have some people who can implement it.

Leo: Love to find out, yeah.

Steve: My guess is, I mean, I'm excited that this has already taken off beyond GRC. And we'll see where it goes.

Leo: Well, I use SSH all the time, for instance, to access my web servers. And it's always the same passphrase. I'd love to use this. And I'm sure this is an easy thing to implement with SSH.

Steve: Well, in fact that is exactly what the guy who posted on GRC is doing. He's using SSH. And after he provides his username and password, he is then prompted for his passcode from the Perfect Paper Passwords.

Leo: And it's a one-time only, you cross it off...

Steve: And you never use it again.

Leo: And because I am - I use these things on hotel networks and so forth. I mean, it's SSL, so I don't worry too much. But it would be nice to have that little extra bit of...

Steve: And it's fun. It is. And you feel like a little - sort of like a spy. Ooh, I've got to type in :ZEY.

Leo: And I can have that in my wallet, and it's perfect. Maybe laminate it, that'd be great. Oh, no, you can't laminate it because you've got to cross it off, okay.

Steve: Well, you could. But, yeah, it's just - and if they get worn and tattered you just print yourself another. Now, you were also talking, Leo, about the inconvenience of what if my PayPal token is at home, I need to log in at work?

Leo: Right, right.

Steve: Well, now, this is a double-edged sword because of course the fact that it's a physical thing means that you would know if it were stolen. You might know if someone looked at it or borrowed it.

Leo: You have to have it for it to work, right.

Steve: But the flipside is you are able to print two copies of your Perfect Paper Passwords, if

you deliberately wanted to put one in a secure location at the office.

Leo: Good point.

Steve: So that you would know - so that if you - but again, this thing is so convenient, who doesn't have their wallet with them?

Leo: Right. Well, that's why I like the cell phone method that my bank uses because I always have my cell phone with me, as opposed to the dongle, which I don't always have with me.

Steve: Right.

Leo: What a great idea, Steve. I think this is great. And what I'm really impressed with is that now it's kind of widely, starting to be widely adopted by others who are implementing it in different fashions. This is great.

Steve: Yeah, again, I'm not claiming that this is some miracle. But it's just a very clean, well thought out, sort of like soup to nuts. In the download .zip I even provide all of the HTML templates and the CSS templates that my site uses that people can see, even though people could grab them themselves just by sucking them off my server, they're there, saying look, I've done all the work. It is easy to print these cards, and they work.

Leo: Steve, you're getting perilously close to open source, dude. Just a matter of time. Steve, it's just - I tell you, what I like about Steve is, not only is he a theoretician, but he's practical, too. It's security applied as well as theoretical. And that is what really makes this podcast, I think, especially useful. Steve, great job. Fascinating stuff. And I can't wait to try implementing it in different ways. I'd love to get an SSH and maybe a PAM module and everything.

Steve: We'll get there.

Leo: Next week, your questions, Steve's answers. And of course more security information, as always. Steve Gibson, a pleasure talking to you. If you want to know more, go to GRC.com. If you go to GRC.com/securitynow you'll find transcripts of this. You'll find Steve's show notes, including the .zip file with all this information. You also get a 16KB version, if you like, for the bandwidth-impaired, of this podcast. Steve provides that all for free at GRC.com, along with dozens of really useful free security utilities like ShieldsUP, Shoot The Messenger, DCOMbobulator, Unplug N' Pray, and on and on and on. Well, and this is one of them, in a way, really. I mean, it's not a download, but you can generate a little Perfect Password page for yourself.

Steve: Well, you can see what it looks like. And I'm hoping that by putting this demo together I will encourage people to adopt it because we need strong authentication, and this is a very strong, secure solution.

Leo: You bet. One other thing we need, we need disk recovery and maintenance utilities like SpinRite. If you've got a disk drive, you need SpinRite. GRC.com. Steve, we'll see you next time. Thanks for joining us on Security Now!.