SECURITY NOW!

Transcript of Episode #181

Crypto Rehash

Description: Before tackling the complete description of the operation of the SSL (Secure Socket Layer) protocol, this week Steve and Leo take a step back to survey and review much of the cryptographic material they have covered during past 3+ years of podcasts.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-181.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-181-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 181 for January 29, 2009: Crypto Recap. This show is brought to you by listeners like you and your contributions. We couldn't do it without you. Thanks so much.

It's time for Security Now!, the show where we talk about - it's the geekily informative show, where we talk about all things security oriented with Mr. Steve Gibson, who is the king geek here. Hi, Steve.

Steve Gibson: Hello, Leo. Great to be with you, as always.

Leo: I have - where did I put it? - my Steve Gibson cap now with a little bill on it. Yeah, in France I got - you know how you wear that little cap?

Steve: Oh, yeah, yeah, my little chapeau.

Leo: Your chapeau. I got one in - my wife gave me one for Christmas. So now I have a Steve Gibson hat to wear.

Steve: Ah, cool.

Leo: Today we're going to need our thinking caps because we're going to recap. Get it?

Steve: [Laughing] Ouch.

Leo: Yeah, that was pretty bad.

Steve: Yeah. What I want to do is I want to do a couple seriously propeller-spinning episodes, one to discuss something we've never discussed before, a cryptographic component known as a keyed message authentication code. And that's the last piece we need for understanding all of the SSL protocol, which is going to be the episode that follows that one. But before we plow in, I thought, you know, let's just sort of step back for a minute and do a cryptographic review, sort of a review of where we've come, what we understand, just sort of an overview. Everyone can - maybe this one they'll only have to read the transcript once to get it all. But I just sort of thought, before we go any further, let's sort of just lay down a little bit of very clear foundation about where we've been.

Leo: Sounds like a great idea. In fact, I was saying before we started, when you mentioned that, I said this is how I learn. I learn - and I guess the best term is "recursively," where you go back again and again, and it gets a little deeper and deeper and deeper each time. I can't absorb it all until I kind of learn some stuff, apply it, and then realize where my gaps are.

Steve: Right.

Leo: That's how you learn computers, I guess, and programming, too, and all of that stuff.

Steve: Oh, I think so. In fact, one of the best methods now we're seeing, when I look at curriculum, programming curriculum, they talk about, like, starting early, getting students to, like, write simple programs, get them on the machine and doing things. They'll sort of inherently sort of explore and mess around and change the code and see what happens. But instead of, like, loading people down with a whole bunch of theory first, where you're sitting around saying, wait a minute, how do I use this, what do I do with this, you get them right in and sort of hook them, and then add successive layers of successive refinement.

Leo: Yeah. Well, we're going to do that.

Steve: Very fractal.

Leo: When we - fractal. There maybe is a better word.

Steve: Fractal learning.

Leo: Well, fractals are recursive, too, aren't they.

Steve: Yeah, exactly.

Leo: So maybe for those of you who have heard our other crypto discussions, maybe it kind of gave you a beginning. But we're going to recap, refresh, revive...

Steve: Review.

Leo: ...review. All right. Time, Mr. G, to - first of all, I guess, before we do - I talked about our recap on encryption. Maybe - are there any updates from last week, any...

Steve: Oh, yes. Well, don't know how long or how often you've had your Macs turned on. But there was a very big - for me on the Mac platform I think it was 76MB - it's a major update, Version 7.6, to QuickTime.

Leo: Oh, yes.

Steve: Which fixed seven different critical remote execution vulnerabilities.

Leo: Oh, you're kidding.

Steve: And one with the MPEG-2 player. So it's very important. It's across all - it's both platforms. Both Mac and Windows versions of QuickTime really do need to get updated. The exploit is just visiting a site. Most experiences will start playing QuickTime without any user interaction. That is, QuickTime is there, and it's ready to go. When you go to a site that's got some QuickTime content, it immediately plays. So there's some concern that this is going to get exploited. At this point there isn't anything known that is exploiting these. But Apple had been informed as early as, like, middle of last year. So these are a little while in coming. So I would say it's absolutely worth taking the time. When I turned my Mac on, it announced that there was the update, which I was expecting it to. But it was so big, and it was, like, 10 minutes before we started recording, I thought, eh, I can - I'll wait till afterwards. But...

Leo: Yeah, I downloaded it on all my machines, yeah. And I didn't realize it was - I knew it was a security update. I didn't realize there were seven remote exploits. That's incredible.

Steve: It's a biggie. Also there was a note that I saw that I thought was interesting, just to sort of warn our listeners. There are illegal copies of Apple's iWork '09 appearing on filesharing websites. It's estimated about 20,000 users have downloaded this. The bad news is it contains a trojan...

Leo: Oopsies.

Steve: ...known as iServices.A, which runs with root access. When you run it, and it installs itself with root access, it downloads - it contacts remote servers, downloads a bunch of additional software to deeply infect your Mac with a botnet.

Leo: You know, this is taking advantage, I think, of the ARD Agent, the Apple Remote Desktop Agent bug that's been around for some time. And the key on these things is, still on the Apple, you've still got to get - you've got to trick the user into installing it.

Steve: Yes, exactly. But, and again, I just wanted to warn people that there are - this Apple's iWork '09...

[Talking simultaneously]

Leo: ...BitTorrent, you nitwits.

Steve: Exactly. Exactly. And I've never mentioned it. I've referred to pirated copies of SpinRite a few times. We have been sent trojans which were called SpinRite, and people anonymously said, hey, I just wanted to let you know your SpinRite trashed my machine, here's a copy of it. And it wasn't SpinRite at all. So it's like, oh, you really do need to be careful when you run things that you get from questionable sources.

Leo: This is a very common way for spyware folks and virus authors to get you to install their stuff. You're installing something from somebody you don't know. This is one of the five rules I tell people on the radio show. Do not accept files from strangers. And BitTorrent is accepting a file from a stranger.

Steve: Yes, oh, exactly. And in fact remember that last week, when I ran Microsoft's - the MSRT tool, and it found seven viruses sitting in my email attachment folder. I had never run them. They were there. There was no one loving them, to start them and let them go. But, and so nothing had gotten into my machine. But some email that I received brought those in.

Leo: Wow.

Steve: So that absolutely happens. You know, we've remarked in the past on this show about how can it be that people don't have Windows Update running, for example, with regard to this Confickr worm, also known as the Downadup, which has infected at this point tens of millions of machines. It's still causing problems. And remember that this is - it's using something that was patched in October. Well, I ran across an interesting discussion about this because five hospitals in Sheffield in the U.K. had more than 800 of their 7,000 PCs infected by this worm. Okay? Why did they have Auto Update disabled? It turns out they had administratively disabled Auto Update, on purpose. Okay. Are you sitting down?

Leo: Because? Because?

Steve: Because, in the middle of surgery, a bunch of their PCs in the operating theater...

Leo: Rebooted.

Steve: ...rebooted and shut down life-critical equipment which was running on Windows. It's like, oh, god.

Leo: The same thing on the TriCaster. The same thing has happened to me many times. Our stream has been shut down. I finally had to turn it off on my machines.

Steve: Yeah.

Leo: This is a stupidity that Microsoft - now, I think they've fixed this, this automatically installing and rebooting. Haven't they?

Steve: Well, it's still an option. I mean, I noticed that on machines that I don't use often - like I've got a little PC set up with a stamp printer and label printer and a little scale for, like, mailing things. And when I turn it on, and that's only every couple weeks, it'll say, oh, I've got updates. And I've got mine set not to auto reboot. But it is a user-choosable setting, like just notify me, or download and notify me but do not install, and then finally is, like, do the whole thing. And so what I - what happens is when I'm shutting it down, it'll give me the option to install updates and shut down. Which is really nice because I can then say, that's what I want, and I walk away and leave it, and it does its thing and then turns itself off.

Leo: I have a download - this is what I ended up doing. Download and notify on all of them.

Steve: Right.

Leo: But never install. In fact, I remember it was doing it with - as we recorded shows it would say, okay, in 10 minutes I'm going to restart. And I'd have to - every 10 minutes I'd look, click, click, click. So, yeah, I think that's a little annoying, Microsoft. But so the default is to do that, I think, as I remember. But the notify - download and notify is good.

Steve: Right.

Leo: Because it says we've got new updates. Let us know when you want to install them. And it will even do it when you shut down automatically, right, if you just - because it puts that little shield up on the shutdown button.

Steve: Yes. And, well, it does in Vista. It doesn't under XP. But you do this...

Leo: No, it does in XP.

Steve: The shield?

Leo: Yeah.

Steve: Okay.

Leo: Maybe that's something new in the Service Pack 3, because we have one XP machine. And I saw that shield, yeah.

Steve: Good, good. Anyway, so for anyone going into surgery, a good idea...

Leo: Ask if it's using Windows.

Steve: ...is to ask, yes, ask your hospital, do you have any Windows machines on the Internet that'll be in the operating room with me? Because I'd like you please to disconnect them from the Internet so they don't update themselves and reboot in the middle of surgery. That's just nuts.

Leo: That's a good point. Why are they on the Internet?

Steve: Exactly. They're on the Internet in the operating theater.

Leo: That's a very good point. That's a dumb thing.

Steve: Yeah, exactly. I mean, they were on the network, receiving updates. It's like, oh, goodness. And I wonder if they might have the Downadup worm, and they're participating in a botnet, and maybe have time to keep your oxygen levels regulated.

Leo: No reason for a computer in surgery to be on the Internet.

Steve: Yeah.

Leo: I don't want it surfing the 'Net. Maybe the doc wanted, like, to look up Wikipedia on how to do this.

Steve: Streaming his classical music while he...

Leo: Betcha, you know what...

Steve: ...opens up your chest.

Leo: ...exactly what it was. He's watching TWiT Live. That's exactly what it is. Stop watching. Go back to work.

Steve: Okay. We have some errata, as well. A poster in the GRC newsgroups, using the handle "ferrix," has been working with the YubiKey at a relatively low level for some time, and responded to last week's discussion of the limited size of YubiKey's static password, which I think was 192 or 176, I think it was 170-something bits. He said it is, although he doesn't understand why the little customizing personalization app that Yubico produces generates these shorter static passwords, he is absolutely sure that it is possible for the YubiKey to generate a full 64-character, 256-bit static password. So it is as long as any passphrase you could ever give to, for example, a WPA router. It ends up hashing it down into, actually into a 128-bit key for AES. So this is twice that long. So it's absolutely as secure as you could ask for. And maybe Yubico will fix their little gizmo. He's got a script which he's been posting links in our newsgroups. We have a GRC Security Now! newsgroup which sort of follows the show, where we have some dialogue about what goes on. And so that's where he's been participating.

Leo: Excellent.

Steve: Somebody else made a note that our recommendation, our Audible recommendation, ["The 7 Habits of Highly Effective People"], was not available to him in Australia.

Leo: Oh, I have to say this every time. I should be much more clear about this. Everything we talk about is U.S.-only. The Audible ads are paid for by the U.S. agency. And any deals or offers we do may be available in other countries, but we do not guarantee. So I don't want say U.S.-only because it isn't necessarily U.S.-only. And the real problem is not Audible. It's the way the book publishing industry, which is a throwback to, like, the 1850s, the way the book publishing industry works, it's hyper-aware of national boundaries. So I'll give you an example: Harry Potter was published in Britain by, I think, Penguin, published in the U.S. by Scholastic. J.K. Rowling, the author, goes to each country and negotiates a different deal in each country. No publisher is international. I mean, Penguin is. It's owned by Pearson, which is international. But each deal is national. So when Audible makes a deal for the recorded rights, it goes to the national publisher. The one in Australia is different from the one in the U.S.

Steve: Wow.

Leo: They may not even have an audiobook. When you got the audiobook of Harry Potter in England, Stephen Fry was reading it. When you got it in the U.S. Jim Dale was reading it. They're different versions. They recorded two different versions. And you can't get one in the other country. So it's just - it's not Audible's fault. It's the way the crazy publishing industry works. It's likely that they didn't have rights for an audio recording in Australia. So I do apol- I'm not saying this to be angry. I apologize. And I should, I guess, from time to time I do mention that, whenever we're doing the Audible ads, these are for - generally these offers are U.S.-only. But I don't say it because sometimes they work in other countries, and I certainly want people to take advantage of them if they do. So, yeah, I'm sorry. We have that deal again. I'll talk about it in a bit.

Steve: Oh, cool. And my last little bit of errata is that I received a nice notice in the mail a couple days ago. CryptoLink trademark has been granted.

Leo: Congratulations.

Steve: So I have the trademark for CryptoLink. That's been underway for some time. I got the domain, CryptoLink.com. Not that I expect really to use it. Everything will be GRC.com. But for a product like this, which I expect to be significant, I wanted to grab the associated domain. So that I've had. But now I also have the trademark for that.

Leo: I should show people my - you get a nice, suitable-for-framing thing, don't you, with the trademark on it.

Steve: Yeah.

Leo: I should frame the TWiT ones.

Steve: Actually my law firm keeps it for me because it's better for them to have it than - I'd be like, okay, where did that go? I'm sure that's around here somewhere.

Leo: I can see mine. It's on the shelf right there. And you get a little, I don't know, I guess you got one probably too, you get - no, I guess this was the corporate seal. I love that, too. I stamp things with my corporate seal. I'm such a kid. I never thought I'd have trademarks and a set of corpor- this is silly.

Steve: Good stuff.

Leo: Yeah.

Steve: And then I do have a fun Security Now! testimonial. This one, the subject of this one was "SpinRite Equals Marriage Enhancement Tool." So I wasn't quite sure what he meant by that. But he said - he's a Security Now! listener. He said, "Steve, I've been listening since day zero to Security Now!, 'As the Worm Turns,'" which I remembered was the first episode we ever did.

Leo: Was it?

Steve: I think we talked about a major worm on the 'Net.

Leo: Oh, wow.

Steve: Anyway, he says, "I owned Version 5 of SpinRite, and I thank you for the upgrade to Version 6. It's my normal process when I get new hard drives installed to run SpinRite at Level 3 and only change levels when I detect problems. So I recently gave my wife a mega-souped-up system, blue LEDs, blue neon, 2+GHz AMD, 2GB of RAM, and a 250GB SATA Seagate drive."

Leo: Yeah, baby.

Steve: "After the OS install, security lockdown, and transferring from backup all of her class documents, our multi-gigabytes of family pictures, et cetera, I ran SpinRite at Level 3, and all was well. Well, that was some time ago. My wife wakes me up one night, stating her Win2K system locked up. And when she tried to restart, the BIOS just sat on the SATA drive detection screen. Resetting the PnP and VRAM nearly got the OS to boot. But the F8 screen progress bar was [chuckily ph] progressing forward, then locked. And it had been several weeks since she last ran the Robocopy backup to server BAT file. So while running at Level 4, SpinRite began listing countless entries of SpinRite-detected damaged sectors dot dot dot, and all the data has been recovered dot dot dot. And after about 24 hours, SpinRite finished the 2.5GB C partition with more than 50 "R" entries," meaning it found problems and fully recovered all the data.

"I reran at Level 5 on C, root, which finished this time with no errors after two hours." Which meant that, you know, SpinRite did fix them during its first pass, and he was rerunning a second pass and found no problems because of the approximately 50 recoveries that SpinRite had done the first time. He said, "My wife was then able to happily boot into her system, check the financial websites that were needed, and all of her files on the D partition were once again accessible. Thank you again, Steve. Your expertly developed program allowed for a Merry Christmas night indeed." And this is signed Christopher A. H.

Leo: And he's a guy after our own heart. He's running Windows 2000.

Steve: Yup. 2K on a super-hopped, speeded-up, all-blue-glowing neon LED machine.

Leo: Isn't that funny. Now, I wonder how safe that is to run Windows 2000. They aren't patching it anymore, are they? Or are they?

Steve: Frankly, I don't think I have a Win2K box around.

Leo: I think if you lock it down, I guess you're probably - you know, when I set up my mom with Windows - she's on a Mac now. But when I set her up with Windows, that's exactly what I did. I set her up with Windows 2000.

Steve: And he talked about security lockdown. So, yeah, you probably move to Firefox, and you don't use IE, and you maybe use Eudora instead of Outlook.

Leo: It's not that, though. I worry about the fact that there may be holes that aren't getting patched.

Steve: Yup.

Leo: You know? And then it doesn't require any effort on your part. It's, you know, like Downadup or Confickr just [sound effect]...

Steve: Gets you. Yup.

Leo: So let's talk about encryption. We've done many - how many shows have we done on encryption technologies? Seems like a dozen.

Steve: I would say over the last four, or three and a half years, we've touched on the topic often because it's, I think, certainly it's interesting. And we rely on it constantly. What we're going to do in several episodes from now is go over in detail, I mean, literally the packet-by-packet operation of the SSL protocol. SSL is - we've talked about it many times - Secure Socket Layer. We've talked about asymmetric encryption, symmetric encryption. We've talked about recently, of course, digital certificates and certificate signing and the recent exploits against the MD5 hash and how that has affected the integrity of SSL. But there is no protocol, no security protocol that any of us use more than SSL. What we've never done is look at exactly how it works. How does it provide these features that we all, to varying degrees, take for granted? So I wanted to do that.

But before we do that, we need to lay a little bit more foundation. And before that I thought, you know, I see people who are - who send notes saying, wow, you know, I had to read the transcripts of that episode three times and, like, slowly, and repeat it to myself in order to understand what you guys were talking about. So I thought - and because, as you mentioned, Leo, we've had so many discussions about various aspects of security, and frankly very little repetition among them, I thought this would be a nice time, before we go any further, to just sort of have a little bit of a timeout and say, okay, hold on, let's step back from the minutiae and from the detail and do sort of a review of the major concepts and components of this that we've talked about over the last three and a half years.

Leo: That's - I'm ready for it. That's fantastic. Because, you know, I've pieced it together, listening to the show. And, you know, I've been here for every episode. But it's nice to kind of get an overview and see if at all the pieces fit together, and then fill in the holes, too.

Steve: One of the analogies that we've used often, I think, that I always get a kick out of, is the Mayberry RFD Opie and Aunt Bee characters because they remind us, by looking at what security is implicit in physical, real-world contact, you're sort of able to better understand what is missing from that implicit security when you get on the Internet. And of course in a physical model, where for example Aunt Bee calls the pharmacist to say hey, I've asked Opie - I guess that was her grandson? - to come over and pick up my prescription. So the pharmacist knows Aunt Bee, knows Opie, recognizes them on sight, recognizes Aunt Bee's voice on the phone, and so there's authentication happening. She dialed the pharmacist, so she's got a good reason to believe that that's the pharmacist on the other end, even though there may be multiple pharmacists, and she may not know them, which one, but she initiated that connection. So she has some reason to believe that that's who she's talking to. Opie has been around town for a while. He knows where the pharmacy is, he knows that the pharmacist knows to expect him and so on. So in the real world, we have a number of things that we sort of tend to take for granted, all of which are missing in this increasingly sort of hostile, security-hostile and predatory environment of the Internet.

One of the things that we need to do is to be clear about the so-called "threat model." That is, what is it that we can do, what is that we intend to achieve, and what things are we not trying to do?

Leo: So in other words, constrain the - not try to do too much, or do stuff that's not needed.

Steve: Well, yes. For example, in our discussions of security we've made the implicit assumption, for example, that it's the communication between two endpoints that we are trying to protect, but that the endpoints themselves have not been compromised. And the point is, if the endpoints are compromised, the game is up.

Leo: Right.

Steve: I mean, if something, for example, a keystroke logger, a keystroke logger will log your keystrokes as you're typing them. It doesn't matter if, once they get on the wire, they're authenticated and secured and boy, you know, nobody can get them during transit. Well, they got them beforehand.

Leo: The other reason for that is it's extremely difficult to protect yourself against physical access. I mean...

Steve: Well, yes. And the other example at the other end of this connection is we keep hearing how remote facilities, whether they're, for example, Network Solutions we heard about the other day, and other sorts of attacks at the remote end, like loss of confidential security information. We hear about how that information is getting away. Well, once again, it may have been absolutely secure and authenticated, and we had a fantastic experience hooking up to our banking site, and nobody was able to get the data as it was going from us to there. But unfortunately, it then sits there on some database on the banking servers, where it's vulnerable. So, again, in talking about what it is we're trying to protect, we need to delineate what it is, what the threat model is, and exactly what it is we're hoping to achieve.

We also make some assumptions. We make, for example, one assumption is that there is non-infinite computational power. That is, that there are not literally infinite resources that can be applied. Because all of the crypto that we've been talking about, for example, every single one of these things we've been talking about is subject to brute-force attack. No matter how long the key is, even though 128 bits or 256 bits, that's a lot of combinations, that's two raised to that power combinations. Well, it's not infinite. It's still a number that we're able to write down. And if we had time, we could test them all. One of them is the right combination of bits. One of them decrypts this. It makes you even feel a little uncomfortable to say that. It's like, wait a minute, you mean there's an answer to - there's a way to crack this encryption. Yes. It's you try all the combinations. But the point is, we've made it so difficult, there are so many combinations, that you can't, it's not feasible in a reasonable amount of time to try them all. But it's always worth remembering that it's not perfect security. It's just really, really, really, really, really good.

Leo: Well, there's no such thing as perfect security.

Steve: No, there is.

Leo: Is there?

Steve: Well, there's the potential for it. We talked a long time ago about a one-time pad.

Leo: Right.

Steve: And if you had a one-time pad that was absolutely random, and the other person at the other end had a matching copy, that is, you're not basing it on a key that's generating pseudorandom sequence, it's truly random, and the other person has it, then - oh, and you absolutely never reuse it. I mean, again, there are restrictions. But there's absolutely no way to crack that. Period. Absolutely no way. Now, the only thing you could do would be to, like, guess every character of the message. But then you could make up any message you wanted. You have no way of knowing which is the right message except by the message length, and you could also pad the length in order to throw somebody off. But in fact there were times during various world wars where this approach was used. You know, ping-pong balls were chosen completely at random. Unfortunately there was one instance where they literally were using ping-pong balls, and there was a bias in the ping-pong ball generator, so that caused a problem. But the biggest mistake made is ever using the one-time pad a second time. It's called a one-time pad for a reason, not just a not-often pad.

Leo: Right.

Steve: So - go ahead.

Leo: If somebody - you could still be compromised by somebody with physical access; right? I mean...

Steve: Oh, that's a very good point. Because, again, we were saying that our end points are secure.

Leo: Right. You could have perfect security between points, of course.

Steve: Yes, exactly. And - exactly. Now, another - a couple of other instances where we're assuming non-infinite computational power is with factorization. There are two things that a lot of our crypto depends upon, that is, our public key crypto. One is that it is very easy to multiply two prime - two big prime numbers together. But it is, as far as we know, there is no similarly easy way to unmultiply the result, that is, to factor that big multiplicand into its two components.

Leo: Its constituent parts, yeah.

Steve: Yes. And so much of cryptography, surprisingly enough, depends upon that one simple fact, that it's easy to multiply two big numbers. It's extremely difficult and time-consuming to figure out which two big prime numbers the result was made from. And tons of time and brain power has gone into this, and no one's found, I mean, and there's lots of factoring theory and lots of factoring, like, improvements. But those are, like, so far away from giving anyone the leverage that weakens our crypto that it's holding up very well.

The other interesting sort of like one-way function is that it is also very easy to take an exponentiation, to raise something to some power. It's extremely difficult to go the other way, and that is to take the logarithm, to get the discrete logarithm of something raised to some power, and then you do a modulus function and take the remainder of this modulus function. That's something that, again, we haven't been able to reverse. And that's another sort of fundamental hard problem that we're assuming no one is going to come up with an easy solution to. If they did, that would be bad.

Leo: Yes.

Steve: Because we'd be in serious trouble.

Leo: Well, every once in a while, when people talk about things like quantum computing, they say, "And it's so good, it could factor these big prime numbers, and security would be changed forever." But so far it's not - nobody's been able to make one.

Steve: No, no. And then another thing about the design of a secure crypto system is you want there not to be any single point of failure. That is, you'd like to have, in a communication network of people communicating, you would - if one dialogue between two parties were to somehow be cracked, you would like all of the other dialogues between other groups of parties, even involving the same endpoints, to retain their security in the face of that crack. That is, for example, if the security only involved everybody using a single preshared key, a single static key, then that would be an example of a system not well designed because the disclosure of that single key would not only allow you to crack the dialogue that was your target, but all the other dialogues that were unfortunately sharing the same key.

So one of the other things that we've seen and that we will be talking about shortly in a couple weeks, is this notion of coming up with some key agreement somehow, but never actually using that key for your live encryption. Instead you'd always use derivative keys that have various limited lengths of life, so that you're not actually using the sort of like the root key that you originally came up with. And so if we assume that the endpoints are secure, that is, they've not been compromised because, as we said, keystroke loggers get in before the encryption, database compromise happens after this encryption, and so we're limiting ourselves to this notion of, okay, let's assume that we have control of each end, but we have no control at all of the link between, i.e., the Internet. So that means that our communication is subject to having bits dropped, bits added, bits changed, and even bits replayed, things, packets replayed. And so we need to also guard against this notion of an attacker somehow, like, redoing something.

For example, say that a communication link with a server involved transferring a chunk of money to PayPal. Well, we would like to prevent somebody recording that whole dialogue, even if they can't understand it, and replaying it to transfer the same amount of money again. So guarding against a replay attack in addition to the idea of injecting new traffic, modifying traffic, or dropping traffic is another aspect of what any truly secure protocol would be able to do.

So finally, let's step back a little bit and say, okay, what do we mean by security? Well, security in this context, in the context of this threat model, where we're wanting to protect communications between two endpoints over the Internet in the face of injection, dropped traffic, modified traffic, and replayed traffic, we want three things. We want confidentiality of our communication, that is, we want nobody, no matter what they do, no man in the middle, whether they're able to intercept the traffic and see everything we do, change it in any way, we want them to be absolutely the case that we're going to have...

Leo: I heard Fred Flintstone, so I [indiscernible] server's back up.

Steve: There's Fred in the background, yup. Somebody purchased a copy of SpinRite. Thank you.

Leo: I love Fred [laughing].

Steve: Well, I did leave him unmuted this time because I know you get a kick out of that thing, so...

Leo: It makes me laugh every single time. I love it. I'm sorry. Go ahead.

Steve: Makes me smile every single time.

Leo: Yeah, I bet it does.

Steve: So what we want is we want confidentiality. We've looked at ways, well, okay, confidentiality. We also want to guard against the message that we're sending being modified. So we need to verify the message's integrity. And lastly, we also want to authenticate the endpoints. We want to make sure that at least one, maybe both, are who we think we're talking to. We talked about this with regard to phishing attacks often, and the Kaminsky attack against DNS that has you talking to a wrong server, to certificate problems. So we want no one to be able to hear what we're doing. We want no one to be able to change what we say. And we want to absolutely be sure we're talking to the endpoint we think.

And so if you think about it, all of that is implicit in real-world communications, in the Opie and Aunt Bee Mayberry scenario. In this case, confidentiality in that example isn't required. But the other things that we take for granted about a physical real-world communication are present. And so what we're really trying to do is we're trying to extend that model across the Internet. With regard to confidentiality, we have talked in the past about encryption, the notion of using - of symmetric encryption, where we share some sort of a key, and we use the same key at each end, thus the symmetry, to encrypt and decrypt.

We've talked about asymmetric encryption, also known as public key encryption, because in the general case we're normally keeping one key secret, the other key is private, although neither has to be the case, but that's normally the way it's used. And because asymmetric encryption is so computationally burdensome, we don't asymmetrically encrypt an entire long message. That would just take forever. Instead what we do is we choose a random symmetric key, and we use the asymmetric encryption just to encrypt it. That way that allows us to transport it to the other end, where the public key, or asymmetric encryption, is used to decrypt that. That creates a sort of a transient shared secret, a shared secret that is just going to be used for some length of time. We're able to get it to the other end, even in full view, by using public key encryption. Then it's used in order to symmetrically encrypt and decrypt our communication.

And finally, key management is an aspect of confidentiality, whether we're using, as I just gave the example of using, public key encryption to encrypt a shared secret. There's another approach, which is known as "key agreement," where it's possible for the ends to publicly disclose what they're sharing, yet maintain full confidentiality. So that someone, an attacker, can see a dialogue going back and forth in the air, essentially. Yet even so, someone watching it can't intercept or can't end up with the knowledge of the key which is agreed upon. So those are sort of the aspects of confidentiality, encryption and key management.

The next thing, the second of these three, of confidentiality, message modification prevention, and endpoint authentication, is this message integrity. We've talked about using hashes. Of course we've talked about MD5 a lot, the notion of creating a signature. And there are the SHA-1 hash, there are more modern hashes that use larger hashing results that are increasingly secure. So...

Leo: Yeah, in fact that's what PGP does, right, when I'm using it to sign as opposed to encrypt. It's hashing it.

Steve: Yes, exactly. And so the idea is that we've talked about a hash being a digest of a much larger communication, where it reduces it to essentially a fingerprint, something such that any modification to the original document would end up changing the fingerprint completely. And it is not computationally feasible - here we go again, we're assuming non-infinite computational power. It's not computationally feasible to make a change that results in a deliberate result. Well, a couple weeks ago we talked about this breach in MD5 where, within limits, these cryptographers figured out how MD5's algorithm is not as strong as we hoped or wanted, where they were able to deliberately get an MD5 digest to come out exactly the same as another document's, which allowed them to take the signature from a certificate authority and apply it to their bogus certificate.

So what we're going to talk about in a couple weeks is a type of digest we've never discussed before, which is a keyed digest, that is, where you mix the notion of a cryptographic key with a digest. And that's important for authentication of the person who did the signing. I've always sort of chuckled a little bit to myself when I've seen download sites that give you the MD5 and even the SHA-1 hash for the file you're downloading. That's always seemed dumb to me because, if somebody was able to put a fraudulent file there, well, then they're also able...

Leo: To change the hash.

Steve: Exactly.

Leo: I never thought of it that way.

Steve: Exactly. They're able to say, you know, here's the MD5 and the SHA-1 for the file. And so you think, oh, good, I'm going to download that, and I'm going to check the MD5. Well, if the file's bogus, then so is the hash. Or it could be because they've obviously compromised the server.

Leo: I guess the idea is that you put the hash on a website, but you're getting the file from an FTP server. So you'd have to compromise two different locales. At least when it's done in open source I think that's the idea. But you make a very good point. Trust no one. That's the Trust No One model.

Steve: Well, it's like the email that says "This email has been scanned and verified by an antivirus." It's like, well, if I were going to write a virus, that's the first thing I'd have added to the end of the email.

Leo: Sure, that says that everywhere, yeah.

Steve: Exactly. So we're going to talk about a way of keying these digests in order to create essentially an authenticated signature, which we don't have at this point because, again, imagine that a communication were going across the wire, and it was signed, even by a really good hash, like by SHA-256, which as far as we know has no weaknesses. Well, some bad guy, since the algorithm is public, a bad guy could replace the message with his own and sign it with SHA-256. And so it would get to the other end, and if the digest were checked, it would match. I mean, it would be properly signed. So this notion of coming up with a keyed digest is one aspect. It's like it's the final thing we haven't talked about that we need to discuss before we get into how, in detail, SSL gives us all the things that we've just been talking about.

Leo: Right, right.

Steve: And the last thing is endpoint authentication. That is, we have confidentiality and protection against message being modified and then endpoint authentication. And that we've covered extensively in the last few weeks when we've been talking about certificates and the way we have a chain of trust which is anchored to a root certificate authority, so we're able to follow the chain all the way to the certificate. And given that every link in the chain is trusted, and the certificate root is trusted, then we can trust the certificate that results to verify the identity of the endpoint.

Leo: I've always thought the chain of trust is kind of an elegant concept.

Steve: I really like it, yeah.

Leo: Yeah. All right. So you've covered a lot of ground here.

Steve: Well, so those are, yes, those are - I wanted to sort of review. Those are all the elements of crypto systems that we've talked about. And the things that we want to achieve, that we will in two weeks, after next week's Q&A, we're going to talk about keyed digests, essentially, how we introduce the notion of who signed this rather than just using a common, fixed algorithm that always results in the same signature. That's the last piece we need before we talk about how SSL actually delivers all of this for us, every time we go to a secure website.

Leo: Very cool. You know, in the security news there was one thing I forgot to mention. Did you see that the My.BarackObama.com site had been hacked? It's a public site that allows people to create groups so that they - it was used during the campaign for fundraising and for organizing. And as with any forum, if you give people access to it, they could put links to trojans and stuff like that. So I guess it's not really that it was hacked, but just that there were - people were putting trojans up on there for people to download. Isn't that terrible? But it's a reminder of what we said at the very beginning. Do not accept files from strangers. And when you see an executable, whether it's on a forum or on a BitTorrent site or anywhere, just stay away from those. Those things are dangerous.

All right. I feel ready. I feel my brain has grown. Everything you said, I understood.

Steve: Isn't that great?

Leo: I understood it.

Steve: This is stuff we've all talked about before. I just wanted to kind of gather it all together in one place so that we have a common glossary, and to move forward. And in a couple of weeks we're going to tackle, finally, how all this fits together into a complete crypto system, and talk a little bit about the history of SSL. It turns out that Version 1 of SSL was not very good.

Leo: Oh, really.

Steve: They used our old friend RC4.

Leo: Right.

Steve: This was from Netscape.

Leo: Netscape, yeah.

Steve: Of course RC4 is the pseudorandom sequence of bytes which just used XOR. And they only protected it with a CRC, with a standard checksum, which meant it made it very prone to being manipulated. Fortunately, that never got out in the world. It was replaced by Version 2 very quickly.

Leo: They also offered 40-bit and 50-bit, 56-bit, I think, encryption.

Steve: Yeah. Well, there were - 56 was of course DES encryption that was there. And that's a good point. One of the things that they took into account, which the original WiFi spec that also uses 40-bit encryption took into account, was U.S. export restrictions.

Leo: Right.

Steve: Until 2000, the year 2000, I mean, really a problem. It was nutso that, I mean, here all this is in the public domain. All of it's in magazines and on the 'Net and fully available.

Leo: Oh, yeah.

Steve: The U.S. government is saying, oh, no, this is - they were calling it "munitions."

Leo: Munitions, yeah. That didn't last, though. Remember the hacker who put the code on his T-shirt and said - and got on an airplane? They just couldn't - they couldn't protect it. The secret had gotten out.

Steve: It was just nuts.

Leo: And they just couldn't protect it. So fortunately we all use 128-bit secure SSL, using - it's RSA now; right? Is that what they use instead of RC4?

Steve: Well, that's what we're going to talk about in a couple weeks. We're going to go over all of that because the way this has evolved is interesting. And we now have all the tools in our toolkit for understanding this.

Leo: Wow, I didn't mean to do a preview. By the way, next week we do answers to your questions. So I want you to go to Steve's site, GRC.com/feedback. If you heard something today, or you've heard something on another show that raised an issue, a question, a lot of times we get great ideas, too. So don't hesitate to go there and say, hey, what about this, I think this might work, or I've got a better way of doing it, or whatever you have to say. You're full of it, Leo. That's okay, too. GRC.com/securitynow. That's the place to go. By the way, while you're at GRC, man, we've got some great stuff, Steve's got some great stuff there for you. Not only all the podcasts going way back to Episode 1, "As the Worm Turns," both in 64K full quality as well as 16K for the bandwidth impaired. He's got transcripts - you don't have transcripts for every show, do you?

Steve: Every show. I had Elaine go back, and we caught up from the very beginning. I said, look, let's lay down a foundation here. So yes, every single show.

Leo: Wow. I would really like to start doing that for other shows. We've got to put transcripts out because that just makes it easier to find stuff.

Steve: I will say the wiki stuff that's been done is just spectacular, Leo.

Leo: Thank you to our wikiarians, wikarians. It's wiki.TWiT.tv. It's an official wiki. We put it up on our servers. And it's using MediaWiki. So if you've ever edited

Wikipedia, please go in there, create an account. We are all invited. And really the community has stepped up, and we've got great show notes there now. As you're talking, Steve, they're typing. They're going for it, man.

Steve: That's a little freaky.

Leo: Oh, man, they do such a good job. So thank you to our wiki editors. They're just - and anybody could be one. That's the beauty of a wiki. So, yeah, TWiT.tv, wiki.TWiT.tv is the place to go for show notes. Steve's got his complete show notes, as well, on his site. And of course once you get to GRC.com you're going to find all sorts of great software, ShieldsUP! to test your router, DCOMbobulator, Wizmo. Somebody asked in the chatroom, I need more instructions on Wizmo. Is the ReadMe not enough for some people? I think it's all there.

Steve: It's all there. I mean, and the page, go to GRC's Wizmo page because I've got, like, a paragraph for every single verb. So there's much more on the website than there is in the app itself.

Leo: Ah, okay. So read the website. That's the thing. Wizmo is cool.

Steve: Yup. And I will mention where my time at the moment is. Actually it's a couple things. I have successfully built three of the little PDP-8 kits. That was over the weekend I did that.

Leo: Now, do you have to solder those, or you just snap them in?

Steve: Oh, it's major solder.

Leo: Oh. You built three of them?

Steve: I built three of them. And it's amazingly fatiguing because I have got this little wacky magnifying headgear that I wear that allows me to see really close. But like I'm holding my breath as I'm touching the soldering iron and the solder to every single one of the little legs.

Leo: Oh, man.

Steve: I'll have one with me next week and hold it up for our live video people to see.

Leo: That's so cool. You built three of them.

Steve: Yeah.

Leo: Why did you build three of them?

Steve: And they came right up, responded to the console.

Leo: You're kidding. Oh, that must be a good feeling, when you build something like that and it works.

Steve: Yeah, yeah. Although it's, I mean, it's advanced kit construction; but all the pieces are there, and all the work has been done by Bob Armstrong, who designed it. And probably about a month from now the front panel kits will come, which are the front panel with all the switches and lights and things. And so those get - I'll build those, and then mate the two boards together and have some blinky lights.

Leo: You are - we've got to do a show on, you know, just I don't know what. You program it or something, I don't know.

[Talking simultaneously]

Leo: ...a separate sidecar show. We'll do a...

Steve: I'll do it. I'll have to write a program specifically to make the lights all blink in a good way. And then where I've been in coding mode, I'm working on bringing what used to be called DNSRU - it's a utility from 2002. I found it was dated 2002 when I opened up the source code. So it's seven years ago. And it stood for DNS Research Utility. I was experimenting with a whole bunch of things about DNS. But it's always been like the secret favorite of the denizens of the GRC newsgroups. It expired, and so you had to hold down a shift key when you started it in order to get around the expiry notice. But it's, as far as I know, it is the most comprehensive benchmark ever created for DNS. And now that OpenDNS has come to the fore, and there are a bunch of other public DNS servers, it'll be very easy for people to run this and compare their ISP's DNS servers, or whatever DNS servers they're using, to a bunch of others. And so it gives you statistics and performance and characterizations and just a whole bunch of stuff. And I'm in the process of finishing it, which is like it's the final piece of this whole DNS region that I've been assembling for GRC, which started with the whole check your DNS servers for their spoofability. So it's all coming together.

Leo: Very cool. So this is a lot of stuff Steve's working on. Again, GRC.com. And it'll all be unveiled there. And let's not forget, there is something called SpinRite there that you just should have a copy of.

Steve: That makes it all possible. It makes everything else that I do possible.

Leo: You just ought to have a copy of that. That's at GRC.com. Steve, thanks so much. Great fun. I felt like I learned something, as always. And we'll be back next week to answer questions. And I'll see you then, Steve.

Steve: Talk to you then.

Leo: Bye bye.