Modes of Encryption

Description: In preparation for a deep and detailed discussion of Secure Sockets Layer (SSL), Steve and Leo first establish some formal crypto theory and practice of encryption operating modes.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-183.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-183-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 183 for February 12, 2009: Modes of Encryption. 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 that covers all your security needs - privacy, online security, encryption. Here's the man who put it all together, Mr. Steve Gibson of GRC.com. Good morning, Steve.

Steve Gibson: Leo, let's do a podcast.

Leo: Let's do - it's a beautiful day. Let's do two.

Steve: It's a beautiful day for a podcast.

Leo: Yeah, it always is because we're inside, actually.

Steve: Yeah.

Leo: And I've got all the windows closed. So I don't even know. Day or night, it's like Las Vegas in here.

Steve: What was it I heard happening during - I picked up a little bit of it during Mac Break Weekly, some modifications you're making to the structure or something?

Leo: The structure.

Steve: Did something fall down? You had to post a bond?

Leo: What? I probably was joking around.

Steve: Oh, okay.

Leo: Did we have to post a bond? I don't think so.

Steve: Sounded like some modifications made to the cottage, anything about that?

Leo: Oh, I was joking around with the Giz Wiz that we'd moved the cottage six inches to the left.

Steve: That's what it was. It was...

Leo: That was a joke.

Steve: It was while I was watching the playback of Giz Wiz.

Leo: Yeah, yeah, yeah. I was just joking around.

Steve: Oh, okay.

Leo: Never believe anything you hear on the Giz Wiz.

Steve: Oh, no.

Leo: It's all pretty much made up, yeah.

Steve: Yeah.

Leo: So today we're going to talk about encryption.

Steve: Once again. It's obviously a continuing topic because it's hard to argue that there's anything much more important to security than issues of encryption. We're heading toward a podcast where we're going to really clearly, thoroughly discuss the number one most used encryption protocol, I mean, like, that there is, that all of us use, which is SSL. And I had said that I was going to - this week we were going to talk about so-called "keyed hashing" algorithms. But in laying that out and sort of getting it, like, put together, I realized, wait a minute, there's something - there's a little bit more foundational stuff we need that leads into that, that we really need to cover first, and that is also necessary for discussing SSL protocol.

So this is - I guess the title of this week's would be "Modes of Encryption," which is something we've touched on a little bit, but never really talked about. And there's also so much security news this week that I thought, if I tried to cram all that into one, it would just - I don't know. People would fall off, I mean, they'd get to work and still have podcast left. They'd fall off their stair climbers. It would just be going too long.

Leo: You're doing this for our protection.

Steve: I always have our listeners in mind.

Leo: Thank you very much. Yeah, sometimes I find that there's a lot to digest in a show. And...

Steve: That ball you're bouncing on would end up deflating by the time...

Leo: That's why the transcripts are great, and people, you know, it's a podcast. You can go back and listen again and again. This show especially, even though we do cover security news and so forth, most of the content is really timeless. And today's will be a good example. The topics we're going to talk about are things you'll need to know about this stuff for a long time to come. But before we get going, I know there's some errata, and there's some security news. And I think Tuesday was a Patch Tuesday. I don't know, I didn't see any patches this...

Steve: Oh, baby. That's where a lot of it comes from. Yup.

Leo: Oh, all right. Well, we'll talk about that in just a second. So let's, I guess, get the news and errata out of the way. And then...

Steve: Yeah, out of the way, Leo...

Leo: First let me get the box over your face out of the way. Okay, now.

Steve: This is a major feature of our podcast.

Leo: Out of the way? This is what people tune in for.

Steve: Everyone looks forward to what new nightmares have surfaced in the last week. And in this case of course we are just past the second Tuesday of the month. So Microsoft has dropped another load of bits on us.

Leo: Yes.

Steve: They're, oh, you know what I meant to - I did the update specifically so that I could do MRT and see - confirm that it did say February. And there, sure enough, MRT. I did MRT from the Start=>Run dialogue, and up comes Microsoft Windows Malicious Software Removal tool, February, 2009.

Leo: Excellent.

Steve: Confirming that all the update happened, and I do have the most recent version of that. And of course it would have done a quick scan after the reboot, and then you can optionally do the deep scan and go away for a long time. Actually, you have to go away for a long time when you download this month's updates.

Leo: Is it that big, really?

Steve: Oh, my god. Well, first of all, the big bad news was that IE7 has a critical problem that affects XP and Vista, but not IE5 or 6. So if you're using IE5 or 6, you don't have a problem. If you are using - if you are now up to IE7 - which of course is the current browser, and they've got 8 in beta. If you're up to IE7, then there is a, just by going to a site that's malicious, you can have your machine compromised. Or, since Outlook and Outlook Express by default use the IE viewer, if you click on a link in email that you receive, that can do the deed, as well. So there were a number of critical problems that were fixed in IE7. So that's been fixed.

There is a new version of the MSRT, as I mentioned. And that was about - a little less than 10MB, about 9.8MB. Then there was this cumulative security update to IE7 is what they've called it. That was 8.6. There's an update rollup for ActiveX kill bits. And remember from us having discussed that before, the kill bits are a feature that Microsoft added to prevent ActiveX controls from running in IE because that's one of the major exploit vectors is IE, you're able to script a page in IE that invokes ActiveX controls. And they can be any ActiveX controls in the system, even those for which it makes no sense to run them on a web page, which has been a source of many vulnerabilities and security problems in the past. So Microsoft is now doing a continually better job of deliberately setting kill bits, as they're called, for ActiveX controls that should not be allowed to run

under an IE context or by a web page because it just makes no sense to do that. So that's been updated.

There were - I'm still using Office 2003. So I can't speak to what may have happened with 2007. But for me, there were two updates that affected Office 2003. Then the big mother was a .NET Framework update. And it's 250MB.

Leo: Oh, please.

Steve: So it's one of those...

Leo: Now, is this required? Because .NET didn't used to be required; right?

Steve: Well, and technically it's still not. But it's one of those things where you'll run across applications. And it's going to be increasingly the case in the future where they say "Requires the .NET platform." And so there's - we're now at 3.5 is the current version of .NET. And so this was the .NET Framework 3.5 Service Pack 1, and the .NET Framework 3.5 Family Update, whatever that is. But together they were 250MB. They downloaded at no great speed and then took forever to install.

Leo: What if people - they couldn't have done this until everybody had broadband. What do people, you know, 57 percent of the nation has broadband at home. But that means that there are 20 percent or something that don't have high-speed Internet.

Steve: Can you imagine.

Leo: What do they do?

Steve: With a modem.

Leo: What do they do?

Steve: Yeah, every four weeks, gulp, here comes another one.

Leo: So cumulatively this is probably almost a third of a gigabyte download from Microsoft.

Steve: Yeah, you're right, you're right.

Leo: Their bandwidth bill must be astronomical.

Steve: Yeah.

Leo: I hope they're using P2P. You know, somebody's laughing at me because I said 57 percent and 20 percent. But remember a lot of people aren't online at all. So that's where that number comes from.

Steve: And I know...

Leo: I think dialup is only 14 percent.

Steve: They're not listening to us, either, Leo.

Leo: That's right.

Steve: Cannot reach - cannot reach them.

Leo: Well, we make a 16KB version for dialup people. We do.

Steve: Bandwidth-impaired, as you so delicately put it, yeah.

Leo: I just read a stat. That's about 14 percent of the population now. But those people are out of luck. They're not going to be updating.

Steve: Yeah. Well, they also don't have security problems, do they.

Leo: Well, do they?

Steve: Not if they're offline. As long as they lock their door.

Leo: So if they have dialup, and they are surfing to these malware sites, you bet they have security...

Steve: Oh, oh, oh, I thought you meant completely out.

Leo: Oh, no. What does that leave left? The 30 percent who are not online at all. Maybe not 30 percent. 20 percent are not online at all. Those people, we don't have to worry about them.

Steve: Well, we also had a Firefox and Thunderbird update. And SeaMonkey, for that...

Leo: I like SeaMonkey. Mozilla, yeah.

Steve: Yeah. Firefox needs to be at 3.0.6. So you'll want to make sure that you got that. And Thunderbird needs to be at 2.0.0.21. And that fixed six flaws, some of which were critical, meaning that even in the Mozilla parlance, doing something innocently can cause your system to be remotely compromised, a remote code exploit. So you'll want to make sure that you're updated there. And I picked up on another little thing that I thought I would just mention. It's sort of obscure. But VNC is a remote desktop application that I know many people use, and probably many of our listeners.

Leo: Oh, yeah.

Steve: There's a client-side vulnerability in both UltraVNC and TightVNC, which are probably the top two most popular versions of VNC, such that if you went to a malicious server - which I think is unlikely because normally you're going to your own server. But if a VNC client for whatever reason were to connect to a server that was misbehaving, there is a way that such a malicious server can take over your machine remotely. So again, I think it's unlikely to happen. Normally VNC users have, like, they're running the server on their system at home so that they're able to, if they're out roaming around, they're able to connect to their desktop at home, which is obviously a server you trust because it's yours. But I just wanted to bring it up to let people know, if they're running UltraVNC, you want to be at v1.0.5.4; and if you're running TightVNC, you want to be at 1.3.10.

Leo: Okay.

Steve: To be current there.

Leo: Got it.

Steve: And lastly, sort of another obscure, but again the most popular one of its class, there's a download manager called Free Download Manager, FDM. And it has a remote execution vulnerability such that, if you were to download a file, using Free Download Manager, from a malicious download site, it can take over your computer. So I think that's not quite as obscure. That's something anyone using FDM, Free Download Manager, would definitely want to know about and update themselves to the latest version to avoid this because it seems to me it's rather common that you might be going to a site that you don't know you should not trust and say, oh, I'm going to grab this file, and bang, get your machine taken over.

Leo: Yeah, I had DownThemAll!, which is a Firefox plug-in, because I was trying to download a large file, I never could, and I put it on there. And I noticed it was really trying to - wanted to download everything from a page. And I thought, you know, I don't feel right about this. That's not good. It was too easy to download a bunch of stuff. So I just - I took it off. I said, you know, I think it really should be that I explicitly say I want this.

Steve: Oh, yes, Leo. I'm sure that's the case. And in interesting news relating to Windows 7 - I know that you and Paul are talking about it a lot. I've been listening to you guys talk about it. And of course the controversial, from my standpoint, security aspect of the reduced strength of UAC, the User Account Control that we covered in detail when it first appeared in Vista, Microsoft softened the UAC operation so that it wasn't popping up so much. And what they attempted to do in Windows 7 beta - we should remind ourselves that it's beta. And by the way, there will be another one. That was not the last one. And so it's a good thing they didn't release it already as Windows...

Leo: Well, there'll be one more RTM. I mean, this is really pretty much it.

Steve: Right. But they're changing the functionality of UAC.

Leo: They're turning up that UAC, which is really good, yeah.

Steve: Exactly. Because unfortunately they turned it down too far. There are now proof-of-concept code on the 'Net which is able to disable UAC. Essentially it's sort of scripting code that scripts the actions that a user would perform in going into the control panel and using the UAC control in the control panel to turn off UAC. And so Microsoft said, uh, whoops, that's not what we intended, of course. And so what they're doing is they're still going to keep UAC less noisy, which people really want. However, what they're doing is they're specifically and explicitly protecting that particular attack vector. That is, it will be - they're protecting the control panel interface to UAC with a much more - by using a much higher...

Leo: That's a good solution.

Steve: Yes, by using a higher privilege process. And they're adding one more thing that they didn't have, which is, I mean, again, in retrospect, like so many of these simple security mistakes, it makes a lot of sense. And that is, if the strength or configuration of UAC is changed, it absolutely requires manual user confirmation.

Leo: Yes. Yes.

Steve: Which it didn't before. That will be in the next release of Windows.

Leo: See, that's the way to handle this. And you could even use that lower setting as long as it said anytime anybody wants to change this setting you've got to...

Steve: Yes.

Leo: You know, your password or [indiscernible].

Steve: Exactly. And so, I mean, so there's...

Leo: It's like that's a case where I think, even if you're running as admin, you should have to enter your password.

Steve: Yes. Yes. For something that is that critical and security-based and has global effect, you absolutely need to prove that you are who you say you are. And it's not something that you would be doing often. You would not be changing the level of UAC operation that often.

Leo: Yeah, exactly.

Steve: And then the last little bit of security news was a little embarrassment that Kaspersky encountered.

Leo: Saw that.

Steve: For about 10 days one of their main corporate databases was exposed through an SQL, a SQL injection attack. We did a podcast some time ago that explained exactly what SQL injection attacks are and how they work. So essentially they had their database exposed, hanging out on the network, out on the Internet. And there were bad guys who were able to access their database. And it was finally those bad guys, after a week and a half, who said, uh, by the way, you might want to fix that.

Leo: Hmm.

Steve: So Kaspersky stated that - and this was corporate, this was their user, I mean, their customer database that was exposed. Kaspersky said that confidential data was not taken or lost or exposed. And so certainly that's good news.

Leo: Kind of embarrassing, though, when that happens to a security company.

Steve: Oh, yes, indeed.

Leo: That's kind of not what you want to have happen.

Steve: We have some interesting news from our friends at Yubico. They've got a new wiki up on their site and are announcing on their site the YubiKing awards. The YubiKing awards are, essentially, are starting now. The idea is to find the best applications and ideas or online services or developer tools or pretty much anything involving the YubiKey. Anyone who contributes on the wiki, which is where these awards will be, or the submissions will be made, and there's a whole bunch of categories for them, will - anyone who submits will get three free YubiKeys.

Leo: Oh, wow.

Steve: Well, but the winner - three winners will be chosen. 50 percent of the choosing comes from participants in the wiki itself. The other 50 percent comes from a jury that I am leading...

Leo: Ah, cool.

Steve: ...for YubiKey applications of any sort. We will feature those three winners on an episode of Security Now!. They will receive five Special Edition YubiKeys that nobody else in the world will ever be able to get. So very special YubiKeys. And this is something that I discussed with Stina when she was out visiting. Remember many months ago she was down in San Diego and swung by Starbucks and had coffee with me for an hour or so in the morning. And we were discussing her notion of a YubiKing award. And she's like, what can we give people? And I said, how about something absolutely special that no one can ever get any other way? And she said, oh, I like that idea. So Special Edition YubiKeys. And finally, the three winners get to attend this year's RSA Security Conference, with Yubico paying their travel, hotel expenses, and conference fees.

Leo: Wow. That's pretty cool.

Steve: So it's going to be very neat.

Leo: That's pretty cool.

Steve: So the challenge is up. If you go to Yubico.com you can click - there's many ways to get at the wiki. But you can click the Developers Menu at the top of the screen, and then you'll see the wiki there. Also, for any YubiKey owners, their wiki is a really nice example of using the YubiKey. That is, it's easy to join. I did it a few hours ago. You say, you know, sign up, and it asks you for your handle. You are never asked for a password because you don't need one. You're a YubiKey owner.

Leo: Right, right, right.

Steve: And they're doing all the online authentication. So it's really - it's an interesting feeling signing up for something that is more secure than any signup you've ever done before because you're using the YubiKey in its one-time password mode with online authentication. Yet it's so simple because you just put in your handle, and then you touch the little button. And it's funny, too, because you don't even have to click, like, log me in because the YubiKey logs you in. I mean, it enters it. The page sees you enter the key and says, okay, let's authenticate and log the person in. Anyway, it's sort of a neat experience. If you're anyone listening to this who has a YubiKey, I would urge you to go try that experience, if you haven't before, of a really well-integrated YubiKey logon experience using the wiki that's now at Yubico.com.

Leo: I now have to get more YubiKeys. I only have one. One is not enough. I need many YubiKeys.

Steve: Well, I think we've stated that two, one that is in static...

Leo: One static, yeah, yeah.

Steve: Yes, static password mode, and one that is in its original one-time password mode, where it's changing every time. And then you've sort of got your bases covered. And it would be nice to have one YubiKey that could do both.

Leo: Do you have a lot of people, now, if you get the one that's not in static mode, it's going to Yubico's servers, and they're using the Yubico server. But are people starting to implement their own YubiKey servers? Because ultimately that's what you want to do; right? Like Bank of America would have its own key server.

Steve: Correct. And, for example - well, okay. The problem with your own key server is that Bank of America would then need to know what the secret AES key was inside your YubiKey.

Leo: But that's what they do with those cards, the VeriSign cards, or the footballs that are sent out by PayPal; right? That's how that works.

Steve: Except that they're all using VeriSign as their back end.

Leo: Oh, okay, okay. So it's normal to use a third-party back end.

Steve: Exactly. And...

Leo: Oh, okay. So having Yubico be the back end is not a problem.

Steve: Exactly.

Leo: I see.

Steve: Yes, yes, yes. And so, for example, in my own forthcoming VPN product that we've talked about before, CryptoLink, I'll allow a user to do it either way. They could, for example, when they're out roaming around and want to authenticate - actually three ways - and want to authenticate, they could use static password mode for if that's the way their YubiKey was set up. Or they could either have the CryptoLink endpoint that they're connecting to, that would be in this case acting as a server, they could have it know the YubiKey algorithm which is publicly known, and have it also know the private key, which would require that they have sort of, like, privatized their key. It would no longer then work with other back-end services because they would have changed their key in order for CryptoLink to know it. Or they could just have CryptoLink use Yubico as the third-party authenticator for their key, depending upon how they want to operate. So all possible modes.

Leo: And we - and, okay, forgive me for asking a - I don't want to insult Stina. But we trust Yubico. I mean, it's not like VeriSign. I mean, everybody knows VeriSign. But Yubico, maybe people don't know Yubico so well. But does it matter?

Steve: Well, I mean, that's - I absolutely trust them.

Leo: Of course you do. You work with them.

Steve: But remember that one of my acronyms, which is...

Leo: Trust No One.

Steve: ...fundamental to what I'm doing with CryptoLink, exactly, is Trust No One. And no one means no one.

Leo: Right.

Steve: And so I wanted to provide a mode for VPN authentication where there was nobody involved except equipment you own - your client and your server - yet still give you the benefit of one-time password, super-strong, non-keystroke-recordable sort of logon. Well, that requires that you then take your key algorithm private so that, without using a third-party back end, you're able to authenticate. But there is the extremely useful mode, for example, of using OpenID, where you really - you inherently need to trust a third party to do your authentication.

Leo: What is the - okay. Let's say that somebody, not Yubico, but somebody like Yubico was doing this, and they weren't trustworthy. What would be - would it be a man-in-the-middle attack? What would the threat be?

Steve: Okay. The threat would be that...

Leo: They would know your password, I guess. But they wouldn't know your password. They'd only know one part of it because you're using multifactor authentication.

Steve: Anyone who could authenticate, by definition, knows the internal state of your key, the whole internal state. They know the value of the counters, which are incrementing, and they know the secret key which is applied - which those counters are applied against in order to generate the result. So anyone who had that could impersonate your use of that key. So if they knew, for example, your username and a site where you authenticate, they could generate the next series of logins into the future for your key.

Now, at the same time, if they're the people doing the authentication, they don't really need to do that. They just have to say, I mean, they could login as you, see the request come back to them and say, oh, yeah, that's fine. So you are definitely trusting an authenticating agent significantly. You're really - you're trusting their complete disinterest in having...

Leo: In your stuff.

Steve: Exactly, in having any reason for impersonating you. And of course, in the case of a real crypto company, their entire reputation is on the line.

Leo: Oh, yeah, yeah, yeah. And please, nobody, am I in any way impugning Yubico. But this is what we talk about, I mean, is what is the risk of trusting somebody. And...

Steve: Yes. You're exactly right, Leo. I mean, everything about security is understanding what we talked about a couple weeks ago, which is the threat model. What does it mean to have what type of vulnerability, and what is the exposure? So it's absolutely useful to discuss that.

Leo: Okay.

Steve: Also you and I were talking before we recorded, but I wanted just to mention I have here in my notes the Kindle 2 has been announced by Amazon.

Leo: And guess who's ordered Kindle 2s? Just take a wild guess.

Steve: You and I were first in line.

Leo: Because, you know, you can't have too many Kindles. I have two Sony eBook Readers which I don't use anymore.

Steve: I have two also. I've got the 500 and the 505.

Leo: Me, too. And I bought the Kindle. But we use the Kindle. We love the Kindle.

Steve: It's the one. I mean, it absolutely is the one.

Leo: But what we've never liked is the form factor. It's ugly. It's too easy to turn pages. I've never been happy with it. This might improve that.

Steve: Well, it's - what I love about this one is it is thinner. And even though the existing Kindle is not that thick, it is funky looking. So this one is classy looking. They've pulled the page turns away from the corners, so now you're able to, like, slip the Kindle into a number of different cases that use little corner elastic straps in order to hold it in. Their mechanism for holding the first Kindle into that little binder was really strange. I mean, it was kind of clever, but it didn't work very well. So it was often falling out by itself.

Leo: I only used it when I was going to throw my Kindle into a briefcase, and I wanted a little extra protection. The rest of the time, right now it's sitting on my bedside table, and it's just...

Steve: Yeah, exactly. And it was like from a material that didn't want to fold back nicely.

Leo: I didn't even know what it was. It was like - it was pressed paper or something, some weird material.

Steve: Yeah. It was not good. And in fact this new Kindle does not come with one. There is no cover provided by Amazon. If you want one, you've got to purchase one from a number of different third parties.

Leo: Okay. So they don't come with a cover at all. They offer their own leather cover, which is 30 bucks. I bought, you know, and then I looked at there's a very fancy leather cover for 100 bucks.

Steve: Cole Haan. Cole Haan has, like, five or six different versions of those.

Leo: But I bought the nylon one that has...

Steve: Yeah, do you mean the neoprene one?

Leo: Yeah.

Steve: Yes, I did also because I think that would be...

Leo: And it said it has protection for the screen and stuff. So we'll give you a review.

Steve: So you and I will have ours coming before the end of February. And I do like the fact, I mean, I'm encouraged by the idea that it's 16 levels of grey because that would allow them to do a little bit better anti-aliasing and to show images, photos and things better.

Leo: So it's a better screen than the old one.

Steve: It's a better screen. It's faster page turn. They did away with that funky weird LCD stripe along the side, where you had the little roller ball in order to, like, do selections.

Leo: Yeah, I'm a little concerned because I used that. I don't know how - they have a five-way thumb thing now; right?

Steve: Well, they have a joystick. It's like left, right, up, down. And so they move an onscreen cursor around. So they must have improved, like, the real-time refreshability of the screen in order to, like, move an eInk object around the screen. And they also increased the battery life. They're saying now that, if you leave the cell modem on all the time, you can get four days of use with the cell modem on 24/7. And if you turn it off, as I only turn it on in order to, like, refresh and update magazines and newspapers and things for a few minutes in the morning, then you can get as much as two weeks of life out of it. Which is pretty much what I get now. So I guess, I mean, it's obviously how many hours per day.

Leo: They measure it by page turns, not by time so much.

Steve: Right.

Leo: Because it doesn't use any power till you change the page. Although it does use power with that wireless thing, and that's the - yeah, I, like you, I turn that off except when I want to download something.

Steve: And so they've moved that control into the UI. Right now in the first Kindle there were two switches, one for main power and one for WiFi, I mean one for cell modem. And now it's in the UI. So that'll be interesting to see how that is. It'd be nice if you could

just say, turn on until you've checked, and then turn yourself off. So I'm really interested to see, you know, what other tunings and tweakings they have done to the UI. But it does look like a nicer solution.

The problem remains, the number one problem, as you and I were discussing before we began recording, is the price. It's $359. It's not something that people can casually purchase. And as you said, if it got down to 200, 199 - and I said 99 - then it would really, you know, people wouldn't just be knocked over. But when people admire it when I'm in a restaurant reading, and then it's like, oh, my god, they want to see it, and they love it, and it's like they read a lot. Then I tell them it's $359, they're like, whoa.

Leo: Yeah, sticker shock.

Steve: Yeah.

Leo: And you do save money on the books, but not a lot of money. The best ones are 10 bucks.

Steve: Yeah. And I'm finding some of the higher end technical books are still...

Leo: Very expensive.

Steve: ...$45. It's like, ow, you know, it's like, what, for some bits.

Leo: There's no reason. That's outrageous.

Steve: Yeah.

Leo: Now, there's one other feature you didn't mention. It can read to you.

Steve: Oh, yes yes yes. That'll be interesting to see, you know, how that works and what kind of, well, what the quality is of that.

Leo: They must have put a much better processor in this thing.

Steve: If so, it's not using that much power. But you're right, I mean, you're right. I mean, doing text-to-speech is not nearly as simple as just decompressing text onto a page. So you're right...

Leo: So the idea is you're reading along, and you're going to get in the car, you're

going to go to work or whatever, you plug it into your stereo in your car and now say read to me, and you continue to read. I think that's really cool.

Steve: I think, if it works, it could be very nice. But you're right. So, like, you're in the middle of a book, and it's like now you're no longer able to read visually, so it reads to you. It does text-to-speech built in.

Leo: It'll take some getting used to because computer text-to-speech is never quite as good as, say, somebody on Audible.com. But handy to have. Especially if you're reading the newspaper. You know, I get the Times on there. And what a great way to just keep up to date. I'm very cur- well, you know, that's why - you buy it because you're into it. I buy it because I have to review, I really want to review it for people.

Steve: That's just an excuse for you, Leo.

Leo: It's why I got into the business.

Steve: Oh, don't give me that.

Leo: This is why - of course it's - I acknowledge that. But it's the whole reason that 25 years ago I got into this business, is so that I could get this stuff for free. And now I don't get it for free, I buy it, but so that I'd have an excuse, a way to keep up with this stuff. When I first started writing for - in 1978 when I started writing for computer magazines, it was just a way of getting free software. And I've never stopped, really. It was really just more ways to get stuff.

Steve: Well, at least you're able to write it off. I mean, it is a business; it's a business expense. And you can tell your wife, hey, honey, I had to buy this.

Leo: Well, and I admit, I mean, I could go to Amazon and say I'd like a review unit. That's what Andy Inako does. But, see, I don't like to send the thing back after two weeks. I want to keep it.

Steve: Nope, exactly.

Leo: So I have to buy it.

Steve: They're never getting mine back.

Leo: All right.

Steve: And then a last little bit is we are beginning to see ultracapacitors in consumer products.

Leo: Oh, you're kidding. This is the thing we were talking about, the EEStor thing.

Steve: Put in snipurl.com/ultracap1. I decided I would start numbering these because - of course now other people could take them. But this is the first example of an ultracap-based - this is Coleman has an ultracapacitor-based electric screwdriver which charges in 90 seconds.

Leo: [Gasping]

Steve: So you stick it on the base. The little meter goes [sound effect]. And 90...

Leo: Oh, my god.

Steve: Isn't that cool? No electrochemical, no batteries to change, no batteries to age and get old. You use it, you know, [sound effect] being an electric screwdriver until it begins to slow down. Then you stick its rear end into the charger, watch the little meter go back up, and in 90 seconds it's 100 percent full charge. And if you're in a hurry, you can just stick it in for 15 seconds and get proportionally that much charge.

Leo: Coleman's calling it FlashCell. So this isn't the EEStor ultracapacitor.

Steve: No, this is not theirs. There are some other ultracapacitors. And in fact there's also a tactical flashlight, but it's still preorder, not yet available. So I'm not talking about that yet. But there is also a very nice-looking, but much more expensive - I think this is $80, isn't it?

Leo: Yeah, it's 79.99. Now, I wonder how long it will go for on that charge. They don't...

Steve: Good question. You don't know.

Leo: If you could charge it in 90 seconds and it runs for 90 seconds, that's not so hot.

Steve: That's not so exciting. There you just want to put a crank on the side of it and say, okay, I'm just going to go back to the old-fashioned way.

Leo: But that is one of the problems with cordless screwdrivers frequently is that it

runs down. I have one that has two batteries, so you swap the other battery in. Okay, I'm reading the copy here. It's powered by the new supercapacitor technology. With this new technology you can recharge/discharge 50,000 times. It'll last a lifetime. What they don't say, and I've got to find out, is...

Steve: How many screws you can screw in.

Leo: How many screws you can screw. 5.4 volts. 220 rpm. 35 lb. inch torque. They don't say anywhere how much you get per charge, and that's what I'd really like to know. Very interesting, though.

Steve: Yup. I will be keeping an eye on that and let our listeners know as supercapacitors continue to emerge.

Leo: Yeah. Very cool.

Steve: Meanwhile, I've got a great, short, fun little SpinRite success story to share. David, looks like Hoeflein. And I'll spell that for Elaine, for her transcript, in my communication to her. He wrote that his computer would not reboot, not even in safe mode. He said, "An attempted system repair revealed that Windows did not recognize the partition type." Hate when that happens. Everybody hates when that happens. I don't think there's anybody who likes when that happens.

Leo: And it says, would you like to format or eject? I mean, that's it.

Steve: Oh, yes. Let's see, A or B. How about SpinRite instead? So he says, "I went to my wife's computer and bought SpinRite, since everyone raves about it on TWiT. After an hour of running, SpinRite detected a sector that it could not repair, but brought back most of it, and that was enough. Windows rebooted, did its disk check, and fixed a lot of indexes, et cetera, and rebooted again. All is as good as new now. This disk had all of my documents for the past 20 years, passwords to everything, and, well, the loss would have been awful for me. I set my new Acronis Scheduler to do weekly full image backups." And then signed "David Hoeflein, new SpinRite customer."

Leo: So that actually raises an interesting question. When it can't recover the whole sector, does it recover part of the data?

Steve: Oh, yeah. That's one of the - as far as I know, that makes SpinRite completely unique among any recovery utility ever written, is that SpinRite is able to, I mean, it tries like crazy to read the sector. It drops into something called DynaStat mode, which annoys some people because it can take so long because SpinRite, essentially it takes 2,000 samples of the sector. But in doing that, it's actually reading the unreadable data. And it does an analysis of it, working to determine where and what the region it cannot read contains. And very often during that time it gets just enough to perform a correction which allows it to recover all of the data and then work with the drive to get rid of this

bad sector and swap a good one in in place of it. Then it writes the data that it was able to recover back into the sector.

Leo: So it's hit or miss whether that'll be enough to say tell Windows your partition is okay. But...

Steve: Well, but no. But here's the other thing is that, remember, a sector is 4,096 bits. It's 512 bytes, 4,096 bits. Only about 11 or 12 bad bits in a row will prevent error correction from figuring out what they are. That's less than two bytes. So that's enough bits to spoil the entire sector. Normally that's it. Except SpinRite is definitely able to give you the rest of them. And so you're able to, for example, out of 4,096 bits, you can get 4,080 bits correctly.

Leo: Oh, interesting.

Steve: And it's very often, for example, if that was a directory sector or even like a part of the boot system, it might be that those two bytes you can live without, literally, and SpinRite will get the rest of them for you, make the sector readable as is. Whereas before it was completely unreadable, it will be readable with the caveat that, okay, we lost two bytes, but you got all 510 other bytes. And oftentimes that's where the miracles that SpinRite performs comes from.

Leo: Amazing. That's really, yeah, so it's that error correction that's really - once you get the sector recovered, the error correction...

Steve: Well, and the fact that SpinRite will give you the data it could read, even tolerating the fact that it couldn't read at all.

Leo: Right. Amazing.

Steve: And very often that's enough gain to make a difference.

Leo: Hey, mewhoelse in our chatroom referred me to a Popular Mechanics review of that Coleman screwdriver. And Popular Mechanics says, yes, it really does charge in 90 seconds, and you get about 30 minutes of screwing.

Steve: Very nice.

Leo: So that's efficient. I mean, if you use it for 30 minutes, and now it dies, and you put it in there, and a minute and a half later you're ready to go again, that's plenty.

Steve: Well, and I'll tell you, do you have Makita?

Leo: Mine is a Black & Decker, I think.

Steve: Okay. I've got two Makitas. And the problem with them is the normal self-discharge. I don't use them very often. So they're sitting in the garage most of the time. When I do need it, I'd like to have it now. And so but the batteries are always discharged, so I always have to anticipate my need, or wait around for them to get a charge. The beauty of this is, first of all, ultracapacitors have an extremely low self-discharge rate. So if you left, when you were done using it, you left it on its charger for 90 seconds before putting it away, chances are it's still going to have a fully useful charge when you grab it. But even if you ended up leaving it discharged, when you want to use it, you only need to wait a minute and a half.

Leo: I love that.

Steve: And you've got a 30-minute charge now.

Leo: Yeah. I'm actually very tempted to buy one of these, I have to say. That's really cool.

Steve: Just to have the technology, yeah.

Leo: Yeah. Yeah. And the fact that the batteries never wear out is a...

Steve: Never.

Leo: ...pretty good thing, too.

Steve: Yup.

Leo: All right. We are going to take a break, come back. We're going to talk about modes of encryption, kind of more ground-setting for...

Steve: Foundation-laying, yes.

Leo: Foundation-laying for our SSL segment, which is going to come up in a couple of episodes. So let's get the foundations down of encryption technology. What do we need to know, Steve Gibson, to go forward here?

Steve: Okay. We've discussed at length the concept of symmetric and asymmetric ciphers. An asymmetric cipher, also sometimes called "public key encryption," is one where you use one key to encrypt and the other key to decrypt.

Leo: Which is a wonderful technology. I have a - if you go to my website, you could download my key. And people will say, well, wait a minute, I'm downloading your PGP key? No, that's the public key. Anybody can have that.

Steve: Right. It's the key of the key pair that you chose to make public. And again, it's one of the things that's very cool about this is that somebody could use it to decrypt something that you encrypted with your private key, meaning the one that you have not disclosed. Or they could use your public key to encrypt something that they would know only you could decrypt. So it works both ways. If somebody used your public key to decrypt something, they'd know that it came from you because only you could encrypt something that your public key could decrypt using your private key, and vice versa. So it's handy.

The problem is, it's extremely computationally intensive. The keys are long. They need to be long. They're, like, 10,000, I mean, 1,024 bits. They're like 1K bit long or longer, sometimes 2K bits because the algorithm requires that much bit length in order to get the equivalent security of much shorter keys using symmetric or private key encryption as opposed to public key encryption. So that length requires, and the nature of the public key algorithms requires, much more computation. So it's never feasible to encrypt an actual communication with public key crypto. Technically you could, but it would just be hugely slow.

So instead what's done is that a random number is picked, just out of the air, a big cryptographically strong, really high-quality random number. And that is used as the key to a symmetric encryption algorithm. And then that key, only that key is encrypted using the public key technology. So then so what happens is, the encrypted document and the encrypted key are sent to someone, and they use the public key technology to decrypt the key, which they then use with a symmetric key algorithm to decrypt the document.

Leo: Isn't that clever.

Steve: It's very clever. And in fact what we're going to talk about today is symmetric key algorithms. We've talked about symmetric key ciphers. And of course the famous one that's become very popular is the so-called "Rijndael" cipher that was chosen as the Advanced Encryption Standard, the AES cipher, after much competition among many different competing ciphers. And we did a whole episode some time ago on exactly how Rijndael works.

Just to refresh people about, in general, the way a symmetric cipher looks from the outside, sort of treating it as a black box, you have some number of bits. You can think of them sort of like signal lines, like electrical wires going into this. And they're either - each bit is a one or a zero. And older block ciphers had sometimes, for example, 64 bits was popular. The problem was that, as computers have gotten stronger, the concern has been that there aren't enough combinations of 64 bits to really make the result strong. So modern block ciphers have doubled that to 128 bits. And it's important to remember that when we double the number of bits, we're not doubling the number of combinations. Every bit we add doubles the number of combinations. So when we go from a 64-bit block to a 128-bit block, we're adding 64 bits, meaning that we're doubling and doubling and doubling and doubling the number of combinations 64 times.

Leo: Wow.

Steve: So the total number of possible combinations is - it's computable, but it's really, really huge.

Leo: 2^64.

Steve: It's 2^64 times more than there were before.

Leo: Wow.

Steve: So imagine we've got 128 bits going into this black box, and 128 bits comes out. So the idea is, I mean, that's the Rijndael cipher, or any similar symmetric cipher. And these 128 bits are transformed through the algorithm in the cipher into a different 128 bits. And the nature of the - essentially it's a permutation. It's not like one bit goes in and comes out somewhere else. It's that, for example, if you were to change one bit going in, on average half of the bits coming out would change. And you never know which half because, if you changed a different bit, a different set of bits on average would change. And not always exactly half, but on average half. Or another example of this is if, say that you had all zeroes, but then you turned on five different bits going in. Well, you would end up with about half of the - a half of one's one bits coming out.

The point is that this is for any pattern of 128 bits you put in, you get out a completely different specific 128 bits. Always the same. When you put the 128 bits in, you get the same 128 bits out, given the key. Because the other input to this black box, in addition to the block of data going in, that is, this block of bits going in and a block of bits coming out, the other factor is the key. And keys can range basically, for example, DES was a very popular - the Data Encryption Standard, basically the prior main government standard, DES, that used a 56-bit key. And in fact that was the source of concern over DES was that, gee, you know, before we had computers, or when they were a lot slower, that seemed to be just fine. But 56 bits just doesn't have enough combinations.

So, for example, the Rijndael cipher allows you, and the AES standard allows you to use a key of 128 bits, 192 bits, or 256 bits. Which is just an insanely long key. I mean, already 128 bits is a huge amount of combinations, so much so, I think I remember reading that, if it took you a second to crack DES with a 56-bit key, it would take something like 142 trillion years with the same amount of processing power to crack it with 128 - crack a cipher with a 128-bit key. So, I mean, so that's the difference in key length in terms of just, you know, the actual number of possibilities given binary bit length growth. It's easy to underappreciate what it means to increase the length of these things. I mean, these things get stronger exponentially as you increase their length, every bit doubling the number of prior combinations.

So we have this cipher, this symmetric cipher. We've put a combination of 128 bits in under the influence of a key that's probably going to be 128 bits. And out comes a different pattern. What's cool about this is that it's a one-for-one mapping. Every 128 bits we put in, we get out a different, I mean, unrelated 128 bits. There's no way looking at this to figure out what magic is going on inside this box that gives us this result. In other words, it is a pseudorandom output. But it's always the same, and it's reversible.

So that gets us encryption.

So say that we now - the question is, and the real focus of what I wanted to explain today, was how do we take that and actually do something useful with it? Which is something we haven't really covered explicitly. That is, say I've got a document with multiple pages. How do I take this symmetric cipher, assume that I've got a secret key that I know, that is, we talked about how the key can be known. It could have been encrypted with a public key technology so that when I got the key, the document was encrypted and the key was encrypted. I used the public key technology to decrypt the key, so now I've got the key.

Or going the other direction, say that we have a plaintext document, that is, a document not yet encrypted. I use a pseudorandom number generator to generate a random 128-bit key. Now, that's the key I'm going to use to encrypt the document. And I will use a public key technology to encrypt that key when I send it with the document, knowing that only the person who's got the matching public key to my private key that I use to encrypt the random key used for the symmetric cipher, will be able to do the decryption.

So the question is, I've got my 128-bit pseudorandom key, and I've got this cool cipher algorithm, this Rijndael AES, or any other symmetric cipher. Now what do I do? Well, the most obvious thing to do is take bytes of the document at a time. 128 bits is 16 bytes. Is that right?

Leo: No.

Steve: No.

Leo: Yes.

Steve: 32 bytes.

Leo: No, 16, yes.

Steve: So 32 bytes is 256 bits, so it's 16 bytes. So I take the first 16 bytes of the document, and that makes 128 bits, put them into the cipher, and out comes gibberish. I mean, just noise, nonsense, nothing. But I write them down. Then I go to the next 16 characters, or 16 bytes of the document, put them into the cipher, and out comes, again, nonsense, gibberish, completely different, given that the second set of 16 bytes are different than the first set of 16. So it's just no relationship that I can see between what goes in and what comes out. And then I take the third set of 16 bytes, put it in, and out comes another 128 bits of garbage, as far as I can tell. And I proceed.

So now you would think, okay, fine, we've successfully encrypted the document. And in fact we have. But there is a problem with this that makes the crypto people feel uncomfortable. And that is, any time we put in the same 16 bytes, we're going to get out the same 16 bytes, or 128 bits, of garbage. Well, in other words, even though we don't know what the 16 bytes were that we put in, someone looking at the enciphered, the encrypted result could say wait a minute, here's the same phrase, here's the same

expression.

Now, obviously those bytes would need to be aligned on the block boundaries in the same way. But the point is, there is some information leakage happening. There is something you're able to glean from looking at the pseudorandom noise. Even though it's pseudorandom and it's noise, it's noise with the possibility of a pattern. And that pattern is, I mean, the pattern reflects, one for one, a pattern in the plaintext. And that's not good. You would say, well, they don't know what the plaintext is. But you've leaked some information.

And, for example, in the case of a - say that we were, instead of encrypting a static document we were encrypting packets. And every packet that was being encrypted was using the same key, which was negotiated once at the beginning of the connection. Well, now all these packets are going by, and there's things that are known about the unencrypted format of the packet, like the header of the packet that contains IP and port number and so forth. And so, if you had enough samples of these packets, and you know that the key was the same, and you know that, because the assumption is always that an attacker knows everything that is known publicly about the protocol, they know you're using Rijndael; they know you've got 128-bit blocks; they know you've got a 128-bit key.

The point is that, in cryptography, you define well what is secret, and you define well what is public. And the whole goal is that the only thing we have to keep secret is the key. If we keep the key secret, we can publish everything else that we are doing, and the result is still private. We still have security. So that we clearly delineate what it is that we're requiring for security.

So the problem we've got now with this first approach is that patterns will easily show through our encryption because we've got a nice cipher which takes blocks at a time. But because the same input always produces the same output, just by looking at the output we can see that what we're seeing we've seen before, and that's information leakage. Well, so this simple approach is known as electronic code book, or ECB, algorithm. Because, think of it, a code book traditionally takes some input and gives you some output. It says here's my code book. I look up this word, and I get this word. I look up this word, I get this word. So that's essentially what this is doing. ECB just - it takes whatever you give it, and it gives you something else. But when used as a protocol, the problem is that it always gives you the same thing out for the same thing in. We need something a little fancier.

So the first thing people came up with was something called "counter mode." Or whereas this first one was Electronic Code Book, ECB, the counter mode just has the acronym CTR. So with counter mode, we operate a little differently. We imagine that we have going into the encryption block, instead of actually putting the data to be encrypted into the top of this cipher, instead we put a counter. We have a binary counter which starts at some particular value. We could start it at zero, but not starting it at zero gives us some additional strength. So imagine for now we just - we'll start our counter at zero for the sake of explanation.

So this 128-bit counter feeds into our encryption algorithm. Well, we already know that what's going to come out is pseudorandom data, even though we're putting all zeroes in, then 000001, 000010, 000011, you know, we're basically doing a simple binary progression feeding into the cipher. What comes out is noise. Thanks to the brilliance of a good symmetric cipher, you can even just put simple progressive counts in. And what you get out is just static. There's no discernible pattern. And remember that, again, this is under the influence of the symmetric key, which is also going into this black box. So now we've got noise coming out.

Well, we've talked a lot about the XOR operation, the exclusive OR, where the idea is that essentially one bits in one of the terms of the XOR serve to invert the bits of the other term of the XOR. In other words, if you were to XOR something with all zeroes, you'd just get it back out again. There's no change. If you were to XOR something with all ones, then all of the input bits would be inverted when they come out. So if you XOR something with random noise, this is one of our other, like, cool fundamental principles. You XOR data, good, normal, plaintext data, with random noise. What you get out is random noise. The XOR, I mean, even though it doesn't seem like you've done enough to, like, really encrypt something, if it's random noise, what it's done is it's randomly inverted the bits of your data. And when you do that, your data is gone. They're just like they were as random as if you had random data coming in in the first place. So this...

Leo: But it's reversible. That's the key.

Steve: Exactly. Because, exactly, because since the bits are being inverted under the influence of one of the terms of the XOR, when you do it again, those bits that were inverted get reinverted, which puts them back the way they were. So exactly. It's reversible. Okay. So now we have our counter set to zero. We feed that zero value through the cipher under the influence of the key, and out comes 128 bits of noise. We then take the first 16 bytes of our document and XOR them with the first 16 bytes of noise, and we get more noise. But we get special noise because it encodes, it encrypts the original plaintext. Now we increment the counter to one, and we feed that through the cipher and get a new 16 bytes of new noise, which we XOR with the second 16 bytes of our document. And now we get a second block of 16 bytes of noise. And we proceed with each 16 bytes at a time, with the counter incrementing by one every time.

Well, now it looks like - now look at what we've got. We're using our cipher to turn a sequential count into a keyed sequential system of noise. That is, even though the counter might start at zero and go one, two, three, four, five, under one particular input key we'll get one series of pseudorandom noise. Under a different input key we get a completely different series. But unlike electronic code book mode, notice that even if we gave the same 16 bytes into this system some time later, the counter is guaranteed to be at a different count because it's counting sequentially. So it won't - and it's 128 bits long. It's not going to repeat in the lifetime of the universe. So that means that even the same data being encrypted with the same block alignment will give us a completely different output. We've solved the problem of there being any patterns. That is, we've solved the problem of any pattern that exists in the plaintext surviving and showing up in our cipher text. So we've got an improvement.

Well, this was better. But there was one next stage that the cryptographers decided would make them feel more comfortable because there is still a property that we have which we could improve on. And that is, each block stands alone. There is no interblock influence. That is, for example, nothing that we're encrypting is dependent upon anything that came before. And it would be nicer, even though this seems strong, it would be nicer if a change in the input text that we're encrypting changed more than just its 16 bytes. Remember, since we're doing this right now a block at a time, block of 16 bytes at a time, each block is isolated. Well, it would be nice if changing - if there was, like, more influence with our plaintext.

And it turns out it's simple to do that. All we have to do in order to create that is, again, change our algorithm a little bit. We'll step back from this counter mode and imagine that we sort of go back now to this electronic code book mode, remember, where we're actually encrypting our data through the block cipher to get our encrypted result. Now

imagine that we take the encrypted output from the first block and XOR that with the second block's plaintext, the source data, before we encrypt it. What that does is, that essentially it takes that pseudorandom output from the first block of encryption, and by XORing it with the second block's input, it completely randomizes it, then encrypts it, giving us our second block of encrypted data. And we similarly, we take that and use it to XOR the input of the third block, and so forth. And this has turned out to be - that algorithm is called Cipher Block Chaining, or CBC. And it is the - one of the most popular encryption protocols because it is very fast. An XOR operation is something computers just do, I mean, they've got instructions built in, unless you're a PDP-8, and in that case you don't even have an XOR. But every computer built in the last...

Leo: How do you do an XOR if you don't have an XOR? Can you do it with bit shifting or...

Steve: You can actually simulate it with - I'm trying to think if the PDP-8 has an OR. I know it has an AND. But you are able, there is a series of instructions...

Leo: You just write the macro that does it and use...

Steve: Yeah, exactly. Or, well, yeah, exactly. I was going to say I tend to use XOR a lot because it's there and it's cool and it's free for, like, different things. But I imagine if you didn't have one you would tend not to use it that much. You'd come up with other ways of doing things.

Leo: Right. It's just great that it's reversible like that. That's what's cool about it.

Steve: Yes. And it is extremely efficient. So what the industry has settled on for many of its operations is this thing called - this protocol called Cipher Block Chaining where, again, you encrypt the first block of data, and you take the result of that and XOR it with the second block before you encrypt it, and then take the result of that and XOR it with the third block before you encrypt it. And if you think about it, you write it down, like, get a napkin out, that process is reversible, that is, you can - if you decrypt the first block, then you get - I'm sorry. Well, it is reversible. You take the...

Leo: I'll take your word for it.

Steve: You take the first block of cipher text and decrypt it. But then you take it and XOR it with the - and then you take the second one and decrypt it and then XOR it with the second. It's harder to describe the reverse process. But it's reversible.

Leo: Right.

Steve: But one thing we haven't said is, okay, wait a minute, we're XORing each block of plaintext before we encrypt it, except the first one. What about the first one? And so we would rather not have that one even always be the same. Remember that the electronic

code book approach, well, this is similar to that, where we're just encrypting a block at a time. We would rather not even have the first one not XORed. So we introduce something called an "initialization vector" that we talked about way back in the dawn of this podcast, the so-called IV, because it was used, for example, in various other encryption protocols. And so the initialization vector is a first - it's something in addition to the key which is initially used to XOR the data. And in this case, in this particular mode, you'd like to keep that secret. Because if an attacker knew what the initialization vector was, well, you might as well not have it.

So you use the 128-bit key to key this - to key all of these ciphers in this algorithm. And then you have another chunk, which is the same length as the block length, which in this case is also 128 bits. So it forms another portion of your secret. And that is the so-called initialization vector, which is used to XOR the first block. Once you've done that, then you take the output from each successive encryption and XOR the next block's input with that. And in that way, the point of this is, by chaining these operations, any change in any of the output data is propagated through the entire rest of the document, making it essentially completely - you just have more randomness built up and propagating, and that makes the cryptographers happier.

Leo: Of course it does.

Steve: And then one interesting thing, and this leads us into our next topic in two weeks, and that is, imagine if you didn't care about the output of each of these blocks. You have the CBC, the Cipher Block Chaining. And you don't care about the output of each of these. You merely take it, and you chain it to XOR the input. Well, what you end up with at the end is very much like a digest, a hash, because you end up with a final block which, thanks to the cipher block chaining, is dependent upon all the data that preceded it. And any change in the data changes the final result. And that is the definition of a good cryptographically strong digest, or a hash.

Leo: Well, there you go.

Steve: And there you go. And notice that it's one which is keyed. That is, it's not like MD5 or SHA-1, which are non-keyed hashes, where every time you put the same thing in, you get the same thing out. In this case it's a digest which is generated under the influence of a key. So when you change the key, you completely change the result. And we'll talk about why that's important and what that means in two weeks.

Leo: I can see why it would be important. But we'll do - I think I can see why it'd be important.

Steve: It's very cool. And that's where message authentication codes, MACs, come from.

Leo: Aha. You know, it's funny, you say we talked about all this before, but it all sounds brand new to me. So it must have gone in one ear and out the other the last time.

Steve: Well, it's complicated stuff. And we're in our fourth year, too, so. And that's why I think that some bit of continually bringing some of these...

Leo: I agree. I agree.

Steve: ...older ideas back and refreshing them is also useful, especially when we're going to be going out and pursuing some new territory in cryptography.

Leo: Yeah, no kidding. I mean, it's tough stuff.

Steve: But I think it's neat stuff.

Leo: It is. It's very elegant. And it makes total sense. And without it, I mean, you know, we wouldn't be able to do what we do on the Internet. I mean, that's what makes all the transactions secure on the Internet.

Steve: Yes. Everything we do. I mean, encrypted databases, encrypted communications, I mean, I don't know where we would be if we hadn't come up with this technology because we absolutely and utterly depend upon it in order to - in order for what is otherwise uncontrolled communication, to keep it safe.

Leo: This would be a good one, if you wanted to review the transcript - and I think I do - to go to GRC.com. That's Steve's website, Gibson Research Corporation, GRC.com. And if you go to GRC.com/securitynow all the shows are there; transcripts; 16KB versions, as we mentioned, for the bandwidth-impaired. The podcast is there, too. You can listen to it over again. And that's a really great resource. Plus show notes. We've now got show notes on the TWiT wiki, too, although I wonder what sense they'll make out of this. It'll be very interesting to see. But we certainly have links to all the news stories and everything.

And of course, while you're at GRC.com, do not forget, that's where you can get SpinRite. You see why Steve is so good at this kind of stuff. He gets this low-level stuff, and he loves it, and he digs into it, and that's what makes SpinRite such a great program for disk maintenance and disc recovery. GRC.com.

Steve: And I will remind our listeners that next week is a Q&A.

Leo: Oh, yeah.

Steve: And you want to go to GRC.com/feedback. And by all means, send me your feedback.

Leo: By all means. Yeah, we'd love to hear your questions. Not just feedback, but

questions, too.

Steve: Yeah.

Leo: All right, Steve. Hey, thanks so much.

Steve: Always a pleasure. We will be moving forward for a couple more weeks with some similarly propeller-winding crypto stuff that I think people are going to find interesting and, I hope, comprehensible.

Leo: Very good. Steve Gibson, thank you for joining us. All the rest of you, too. We'll see you next time on Security Now!.

Steve: Thanks, Leo.