Symmetric Stream Ciphers

Description: Leo and Steve continue their multi-episode tour of cryptographic technology. This week they analyze the cryptographic operation of secret decoder rings, which they use to develop a solid foundation of cryptographic terminology. Then they examine the first of two forms of symmetric, private key cryptography known as "symmetric stream ciphers." Two weeks from now, after next week's Q&A episode, they'll discuss the operation of symmetric block ciphers.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-031.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-031-lq.mp3

Leo Laporte: This is Security Now! with Steve Gibson, Episode 31 for Thursday, March 16, 2006. Steve Gibson, it's a great day to talk security. How are you today?

Steve Gibson: Yeah, I'm just great, Leo. Great to be back.

Leo: Yeah. We embarked on Part 1 of our Crypto series, very popular, last week, although there wasn't really a lot of hard technical facts in it. It was really mostly about the issues surrounding crypto.

Steve: Right, although we really got a lot of good feedback from it. I think people liked the idea of sort of talking about, okay, the sociological and moral and ethical aspects of this, rather than just plowing directly into, you know, prime number factorization and all of the technology. And in fact, one thing that we - remember we were, like, brainstorming toward the end there, what different types of applications crypto technology had. One thing that we failed to mention at that moment - and I just sort of, I thought of it, like, an hour later, and I was like, oooh -

was personal storage encryption.

Leo: Oh, of course, yeah.

Steve: Yeah. I mean, like, you know, TruCrypt we've talked about before.

Leo: That's kind of like the original use of it; right?

Steve: Exactly. I mean, and...

Leo: You're talking about encrypting your files, your data.

Steve: Yeah. Or, exactly, like a partition on a laptop, or you want to encrypt a USB thumb drive so that, you know, if you lost control of it - the idea is not for, like, transporting the information somewhere else, but if you lost control of it, what would you do in order to prevent people from having any access to it.

Leo: Now, will we talk about applications like that in this series, or is that for another time?

Steve: Oh, no. Well...

Leo: Maybe the last episode?

Steve: My concept is that we're going to talk about sort of the technology and the modules and how they fit together, sort of the fundamentals. And then, I mean, there's so much stuff that depends upon this. One of the reasons I want to do this before we come back to and wrap up the OpenVPN stuff is that a lot of the technology that I want to be able to talk about depends upon crypto. And, for example, when we talk about TruCrypt, which is a fantastic open source technology, I want to build up a vocabulary so that listeners who've been following along know when I talk about how this stuff works really what it means. I mean, I felt a little uncomfortable, for example, when we were talking about Hamachi earlier on. I wanted to explain exactly why it was so clear that this was secure. But I didn't have the underpinning technology that I knew everyone would be able to get what I was talking about.

Leo: So it's good to come back, yeah.

Steve: Oh, yeah.

Leo: So we'll circle back, and then we'll talk about applications for encryption, and Microsoft's built-in encryption, versus things like TruCrypt and all of that.

Steve: Right.

Leo: But that'll be later on in the series.

Steve: Right.

Leo: Anything else you want to correct or...

Steve: Well, a couple things. I have a couple new pieces of technology on the GRC site. One of the things that's been happening is, you know that we have a feedback form at the bottom of

the SecurityNow.htm page on GRC. Many people do remember to put their email address in the body of their note. And I've always had technology that parses the notes to see if there's an embedded email address; and, if so, basically it emails their note to me so that, when I reply, it just goes back to them. But a lot of people don't remember to do that, and so I get these anonymous postings. Which is fine with me, but they'll ask a question that I'm dying to respond to, but they didn't give me their email address.

Leo: Right.

Steve: And of course we know for our Q&A episodes that we'd like to be able to say, you know, just sort of call out their name and say where they're from. And again, some people remember to do that. Some don't. And of course if anyone wants to be anonymous, I've got no problem with that. I mean, I'm Mr. Privacy. But anyway, so I've redesigned the form at the bottom of the page so that there are now optional fields where I'm specifically saying "name," "email address," and "location." Which can absolutely be left blank if someone doesn't want to fill them in. But I think I'm excited because there have been so many responses that have been anonymous, where the person asked me a question I could have easily answered, but I didn't have their email address.

Leo: So you're not collecting emails, you're not trying to invade their privacy, it's just so you can respond.

Steve: And I make that very clear on the page. I say "totally optional information." And I specifically say their email address will only be used to address their note to me. No record of it is being kept. And that's the case.

Leo: It's a shame, but nowadays you really always have to say that. Anytime you start collecting email addresses, people are...

Steve: Well, and I appreciate it, too. Because, you know, when I go to the site and they need all this from me, it's like, oh, now what? You know, am I going to start getting bombarded with spam?

Leo: Right, right.

Steve: Well, the other thing that I did is we got this really neat note last week. I forwarded it to you because it was from a Security Now! listener, and I really liked it. It's short, so I'll read it. He said, "As a listener of your Security Now! podcast, I had planned to upgrade my old copy of SpinRite just to show my support of the podcast. But when my boot drive died last night, I knew I needed SpinRite 6 for a different reason. Since my old copy didn't support my new hardware, I upgraded, then downloaded Version 6 on a different machine. I burned a boot CD and popped it in the machine with the failed boot drive. A couple of hours later, I'm writing this on the machine with the previously failed drive..."

Leo: Wow.

Steve: "...and everything works great."

Leo: That's nice. That's nice.

Steve: "Many thanks for...." Yeah, I mean, I loved it. And we've got this page of testimonials like this for SpinRite. But, you know, me and my URLs, it's, I mean, even the word "testimonials," I have to - it's t-e-s-t-i-m-o-n-i-a-l-s, I mean, it's got as many vowels as it has consonants.

Leo: Too many letters, yeah.

Steve: Anyway, so what I did was I thought, okay, let's make this easier for people to find because I know that we've got Security Now! listeners who are wanting to turn their friends and other people on to SpinRite because we've received email like that. So SpinRite.info, the domain SpinRite.info takes you directly to that page. And it doesn't matter how you spell "Rite" - r-i-t-e, r-i-g-h-t, w-r-i-t-e." So just SpinRite.info. And that's the URL we can use from now on, Leo.

Leo: Steve, you're learning, you're learning.

Steve: I'm getting there. I'm a little slow.

Leo: Well, that's okay. Nobody, well, actually I shouldn't say nobody's ever accused you of being a marketer. You have been accused of being a marketer, but anybody who knows you knows that you're not very good at it.

Steve: Yeah, well...

Leo: You're too much of an engineer, I'm afraid.

Steve: I love, exactly, I love the technology.

Leo: Yeah.

Steve: Okay, now, another thing that's going on that is worthy of mentioning because this is a hoax that's been around now for about six months, and people keep sending me various links to it. There's this hoax that all computers and laptops have built-in hardware keystroke monitoring. And the hoax is, it's one of those things that's good enough to almost be believable. It includes photos of this little inline keystroke logger, which actually is an inline logger hardware. But the claims are that all laptop and some desktop machines have this built in. And the hoax goes on to talk about how this one guy innocently opened up his laptop because he was trying to change something, and he discovered this little lump in the wire between the keyboard, with a wire going to his Ethernet adapter. And this goes on to say that, you know, it's sending the information out. And in fact some moderately respectable sites have unfortunately been perpetrating this. And, you know, I'm looking at the headline on one page here that says "Government and computer manufacturers caught installed hardwired keystroke loggers into all new laptop computers."

Leo: Oh, please. Oh, please.

Steve: But the problem is, you know, people are sending this information to us, wanting to know, oh my God, what do we do? Okay, it's a hoax. It is not going on.

Leo: It was a poorly crafted hoax because it was pointed out very quickly when people saw the picture. It was just bogus.

Steve: Yeah. But again, you know, people who aren't so technically oriented are going to say, well, you know, maybe this is possible, you know, could this be really happening.

Leo: Well, I'm glad we can lay that one flat. There's a hoax a minute. I go to Snopes.com whenever I get an email like that, that I even - I mean, most of the time I go, yeah, right.

Steve: Yeah.

Leo: But if I don't - if it doesn't - if it is not obviously a hoax, I will go to Snopes, Snopes.com, and it's a great site for those things and debunking them.

Steve: Okay. Now, following that, we have something that is unfortunately not a hoax.

Leo: Uh-oh.

Steve: Ars Technica reported yesterday, and I've confirmed this, that some researchers in Amsterdam have verified that RFID tags can be viral carriers.

Leo: Oy-yoy-yoy.

Steve: I know.

Leo: How would you do that?

Steve: It turns out, if you have a bad, a deliberately malicious RFID tag, and it's read by an RFID reader, as these things do, you can cause a buffer overrun in the backend database which is collecting data, get code to execute of your choice, and because the RFID tags can also store information, that code is able to then update any other non-infected RFID tags, infecting them with the same virus, so it spreads.

Leo: Now, that requires a buffer overflow flaw in the RFID reading code. Do all RFID readers use the same code?

Steve: No. So the point of this, it's not to say that, oh my God, all RFID systems in the world are now bad. But apparently the RFID people haven't been taking enough concern, paying enough attention to issues of security.

Leo: They never thought somebody would attempt a buffer overflow exploit, you know, on an RFID tag.

Steve: Yeah, I mean, it's...

Leo: Which of course they would.

Steve: Exactly. If it's there, somebody will hack it.

Leo: Didn't you know that?

Steve: And so, you know, this goes back to what we've said over and over and over. I mean, new things are scary because they haven't had time to be proven. It's impossible to claim that something new is secure because that's something that only history can judge. And of course security is a hard thing to do. Clearly the RFID people, you know, didn't do this on purpose. They just didn't check all their code because this never occurred to them.

Leo: Well, in fact, if you think about it, I mean, there's very little data in an RFID tag. And you'd think the software to read it would be very simple. It's kind of amazing that something as simple as that has such a serious flaw.

Steve: I'm glad that it has surfaced this early.

Leo: Yeah, right.

Steve: Before we start seeing RFID-laden passports that are, like, infecting each other and all this. So it's a good thing. Okay.

Leo: All right. On with the show, ladies and gentlemen. Crypto 20 - is it 102 or 201? It's been a long time since college.

Steve: I don't know. It's...

Leo: Our second edition.

Steve: Because we're going to talk about secret decoder rings, I think it's maybe 102.

Leo: Okay.

Steve: It's starting crypto.

Leo: Yeah.

Steve: Now, it turns out that - I mean, and literally I want to talk about a secret decoder ring because it's a perfect, easily visualizable model of simple crypto. And we're going to talk about how you can take that and extend it to make it unbreakable. And this is essentially what so-called "symmetric crypto," or private key cryptography, is about. Now, it turns out that the original - I did some research on this, believe it or not, on secret decoder rings. The original rings would use concentric rings. The outer ring was alphabetic, A through Z. And the inner ring was numeric. They would actually use numbers 1 through 26. And so you would set this to some position, the inner versus the outer ring, and then look up the message you wanted to encrypt by looking for the alphabetic character on the outer ring and then look in to the number immediately adjacent to it on the inner ring. That's how those things actually worked.

We're going to modify, for the sake of discussion, that traditional secret decoder ring because the way normal crypto works is that it translates the encryption alphabet into other characters of that alphabet. Now, when I say "alphabet," I mean the entire set of possible characters. So not just, for example, A through Z. For example, in our ring we're going to use a 28-character or 28-symbol alphabet because I want to have a period and a space. So we'll have A through Z and period and space, which are in a ring on the outside. And we're going to have another A through Z, period and space, which are on a ring on the inside. So we have this 28-symbol alphabet. Now, if the rings are aligned so that A is opposite A and B is opposite B, we obviously get no encryption. So, for example, the term for anything that's unencrypted is, sort of by universal agreement, is "plain text," or sometimes "clear text."

Leo: Right.

Steve: But, I mean, even if it's not textual, it might be unencrypted binary stuff which just looks like, you know, gibberish of hex junk. But we still call it "plain text" as sort of a term of art in cryptography. And similarly, after we do something to the plain text to obscure it, to encrypt it, it's then called "cipher text," again by agreement. So if our inner and our outer secret decoder ring, or modified secret decoder rings, are aligned, we get no encryption. So in order to use this very simplistic encoder, you would twist the rings so that they're no longer aligned by some amount. Now, another characteristic of this is that there are only 28 possible combinations, that is, 28 possible keys, essentially, for this algorithm because, as soon as you go 28 times, you're back around to the beginning. But that turns out to be useful, as we'll see in a minute also. And we already know that one of those 28 provides no encryption at all. So if we rotate our secret decoder ring off to some angle, some number of steps, then the process of encoding is looking up, you know, we write down our message that we want to send. And then, for every character, including periods and spaces, since we have a 28-symbol alphabet on ours, we look up from the outer ring, we find the clear text character, and we look down into the inner ring for the matching cipher text.

Leo: And you don't move it during that - for that message. You leave it that way.

Steve: Exactly. Oh, exactly. For the entire message you do not rotate the rings relative to each other. You then run through the whole message, and you end up with this encrypted character

stream. Now, you give it to a friend who - and you have to give them one other piece of information. You need to tell them how many places, we'll just say clockwise, to rotate the rings, the inner ring from its normal alignment, in order to decrypt this message. So your friend basically sets his secret decoder ring identical to the way yours was set and then performs the reverse operation, that is, you know, for every character on the inner ring, he looks at the character adjacent on the outer ring and writes that down and is able to, obviously, reverse the process you used and get back the original message.

Leo: Simple enough.

Steve: Simple enough. Well, it suffers from a couple things. First of all, it obviously suffers from not having many possible keys, that is, as we saw, one of them that is no rotation of the inner ring produces exactly the same cipher text as plain text, so that's not very good encryption.

Leo: Right.

Steve: Then we only have 27 other settings before we repeat. So it's feasible, even if someone got a hold of your message at school, for example, and they weren't a really fancy cryptographer, they could just put the ring in the first notch away from its normal synchronized position and try a few characters of the cipher text to see if it starts to look like English. And then, if that doesn't, they click it to the next position and try it again, and click it in the next position. They've only got 27 positions total. So in a relatively short period of time they're going to stumble upon the "key" that you used for this encryption and be able to decrypt the entire message. So it's not very good from that standpoint. There aren't a lot of possible keys. And we'll be coming back to that because that's where the so-called 128-bit encryption comes in. The 128-bit or 256-bit or so forth, that's the length of the key in binary bits which specifies how many possible keys there are. And of course that needs to be a bit number for the sake of brute force testing, that is, basically rotating our decoder ring through all of its possible positions. And in this case there aren't very many of them.

Okay, now, the analysis, say that you wanted to attack the cipher that we've generated with our secret decoder ring not using brute force for whatever reason. You know, you want to do more of a sort of a standard cryptanalysis of the process. Well, that turns out to be not very hard. If you knew that this was English, for example, or whatever language it was probably encoded in, you could look at a table of the occurrence of different letters in the language. For example, we know that A, E, T, and S occur very frequently. And so with a large enough source message you count up the number of occurrences of each of the cipher text symbols, and you can, you know, any kind of a cryptographer would look at this and immediately be able to identify what the vowels were and probably what S and T is.

Leo: In fact, it's so easy that even some daily newspapers have cryptography, cryptanalysis problems, you know, challenges like this. They're very easy to do. They're like a crossword puzzle.

Steve: Right.

Leo: E-t-a-i-o-n-s-h-r-d-l-u, etaionshrdlu. That's how I know it.

Steve: And that's the sequence of English characters from the most frequent down towards

lesser frequent.

Leo: Right.

Steve: Now, the second clue are character pairs because there's strong association in most languages with pairs of characters, things that either rarely occur or occur very often. And that gives you another clue. It turns out, as you go to longer sequences of characters, you begin to get a little bit lost. There's not that much statistical weight, if you look at eight-character sequences, for example, because there are so many possibilities, and not many of them tend to occur that often. But the use of single-character and character pairs completely cracks our simple little code. Okay. So.

Leo: Yeah. So that wasn't such a good encryption. Were decoder rings used for anything else but Captain Midnight and old-time radio serials? I mean, do they go back to Roman times? How long have they been used?

Steve: Good question. I didn't go that far back. I don't think they were ever taken very seriously because they're so easy. I think they were...

Leo: So obvious.

Steve: Yeah, they're just so easy to crack. Now, let's dispense with the inner ring and just look at the outer ring. So imagine now that we have a next-generation encoder/decoder system. And all we're going to use is just a single ring of A through Z with period and space. Now, in fact, what we know of the prior design was that all we were really doing was finding a character and moving a certain distance, a certain number of character spaces, for example, clockwise. And we were wrapping around the end of the sequence back to the beginning. So, for example, if you had A and you went three characters forward, you'd be B-C-D would be your ciphered version. Or if you were Z, you'd go three spaces forward, you know, dot-space-A would be your ciphered version of that symbol.

So imagine that instead we just have a single ring. And again, our key is just how many spaces we go forward. Well, it turns out that there is a simple thing that can be done to create an absolutely unbreakable code system using this approach. And that is, if the number of spaces we go forward for any given symbol we're encrypting is a random number. And...

Leo: As we've learned before, randomness is really kind of key to creating difficult-to-decipher things.

Steve: Yes. Randomness comes in all over the place. And in fact we'll be talking about the uses for and the need for what are called "cryptographic quality" random numbers. So imagine now that someone had maybe ping-pong balls and drew a number 1 through 28 for how far you could go. Actually, we wouldn't want 28 because that puts you right back in the beginning. 1 through 27 or 1 through 28? That's a good question. I didn't think this all the way through. But so you've got - for me, I think we want 28 ping-pong balls. So you have a basket of these, and you swoosh them around and pull a ball out and write down the number. Put that back in, stir it around again, pull another ball out, write down that number. Put that ball back in, stir it around again, pull a ball out, and so forth. So what you're doing is you're using mechanical randomness to create a string of really very high-quality random numbers. And in fact, this approach was used in early wartime to create unbreakable ciphers. So...

Leo: There's one problem, though. How do you communicate this random number to the person who's going to read the cipher?

Steve: Well, that absolutely is a problem. So let's go through the process first.

Leo: Okay.

Steve: And because all the things that have happened...

Leo: I mean, if you have a ring, and you've got two rings, you can give the guy the ring, and he can do it. But as soon as we start this random process, how does he know what ping-pong balls you've pulled?

Steve: Well, what this is called, it's called a "one-time pad," is this technology. And so the idea would be literally an implementation. You'd get a pad of paper, and you would sit there, or maybe get an assistant to sit there, and pull out balls one at a time, write down the number of the ball that you got, put it back in, stir it around again, and pull out the next one. So basically you create an entire pad of random, I mean, truly random numbers, numbered 1 through 28.

Leo: All right.

Steve: Okay. Now, to use this, we take our existing ring, that is, our A through Z plus dot and space, we take the ring and find the symbol that we want to encrypt, and take the first number written down on the pad and move that far clockwise around the ring and write down the corresponding symbol. Then we go to the next symbol that we want to encrypt, take the next number on the pad, go that number forward, and write down the symbol that we find at the ring that many spaces clockwise around, and proceed for the entire message. Now, what's...

Leo: Now we've got a great and impossible-to-decode message, even by the recipient.

Steve: Well, but that's the point that is so critical and what I wanted to draw this illustration for. It is impossible for anyone to decrypt this message. I mean, it turns out that what we've done is we've taken very, very non-random text - we already know that the source text can be analyzed all kinds of ways in order to determine, you know, frequency distributions and characteristics and all that. It turns out, though, and this is a critical component of cryptography, is that if you mix your clear text message with truly random data, there is no force on earth - and I don't mean like what we're talking about normal crypto where it will take X number of machines a certain length of time to get this done. I mean it cannot be done. When you mix a message with randomness, and this just counting around a ring is a completely valid form of mixing, where you mix content with something random, no analysis can reverse that process. And that's what's so cool. I mean, none.

It's a key concept for people to kind of get their minds around is that it just, even though your message has affected these random numbers, because they were random in the beginning, pulled from these ping-pong balls, the result is every bit as random, even though there is a way, if you knew the original pad, to reverse the process. And so, for example, in order to decode this, of course, the decoding process is pretty clear. Somebody who had the pad that was originally used, or I should say a copy of the pad that was originally used to encrypt this

message, would take one of the cipher text symbols and look at the first value on the pad and go counterclockwise back to the original symbol that was used and write that down, and then take the second cipher text symbol, look up the second number on the pad, go counterclockwise, and get back to the original text.

Leo: So they need a copy of your pad, is what they need.

Steve: Yes. And in fact this approach, for example, was used in submarines and to encrypt the most important data that the government has ever produced back in the old days.

Leo: And it would still work even with high-tech computer analysis today. It would still be undecipherable. Am I right?

Steve: It is more unbreakable than anything else. I mean, absolutely, completely unbreakable. And that's the key.

Leo: So this is a good system. Who don't we use it anymore?

Steve: Well, because of the point you brought up very early on here, is the key to decrypting it is having a copy of that original one-time pad. Now, the term "one-time" talks about one of the two great weaknesses. First of all, you've got to get the pad to the person decrypting it. Now, back in wartime, what would happen is literally there were people with bingo machines or ping-pong ball machines or whatever, reaching into them, pulling out numbers, creating pads. I mean, this is, you know, in the early days of the NSA. Actually it's probably pre-NSA. But the idea, I mean, this was actually done.

Leo: Right.

Steve: And so submarine captains would go to sea with a briefcase full of these sheets of paper.

Leo: They had a pad. They had the duplicate.

Steve: Exactly. They had a copy. I mean, it was, you know, the whole courier with the handcuffed-to-his-wrist case thing because, if this got away, this would completely allow the code to be broken.

Leo: Right.

Steve: So the idea would be you needed what we call a "secure channel." You need a secure channel to get the one-time pad to its destination. But once there, you are then able to, I mean, literally during the middle of the day, just say the numbers out in the air for anyone, I mean, you could publish them on the front page of The New York Times. Doesn't matter at all what you do because nobody can crack this code who doesn't have a copy of the pad that was used to produce it. No analysis.

Leo: But that's the weak link is that you have to share. It's a shared key system.

Steve: Yes. Well, it is a shared key, also known as a private key. And in this case it's also a symmetric key, meaning that the same key, in this case a pad, which is used to encrypt the message, is used to decrypt it. That's what a symmetric key means. It means that the operation, even though you may do something different technically for decryption, for example, in this case we went counterclockwise around the wheel to decrypt and clockwise around the wheel to encrypt. So you might do something different, but you're using the same key. So it's considered symmetric cryptography.

Now, I said there were two weaknesses, one being that, again, that you need to have a so-called "secure channel," in this case, you know, a courier who you trust, and you make sure there's no way for these documents to fall into enemy hands. The second weakness is you absolutely cannot ever re-use the pad a second time.

Leo: Why is that?

Steve: Any pages you use have to be torn up and burned, I mean, securely destroyed. The reason is that, as soon as you have two encrypted messages which were encoded or ciphered using the same pad, cryptanalysis will allow you to basically separate the messages from each other and decode the message. There are ways you're able to slide two messages against each other and non-random peaks occur which cryptanalysis is able to then determine what was really going on prior to the cryptography. And it turns out that some of our enemies who were using one-time pads didn't recognize - and this is, you know, decades ago - didn't recognize the vulnerability of the system, and we cracked their codes because they were using pads more than one time.

Leo: Wow.

Steve: And so our analysis would check for that happening, we found it was happening, and we were able to determine what the original pad's contents were. Once we had that, we could then easily decrypt all the messages that were encrypted with that same pad.

Leo: Wow. That's pretty amazing.

Steve: So, I mean, it's really neat. And it's interesting, too, because you go from absolutely no force on earth can decrypt this to it's trivial to decrypt it if you make the mistake of using it twice. So it absolutely has to be only used once.

Okay, now let's go the next step in our development of this. Rather than using a ping-pong ball machine and true random numbers, we're going to use a mathematical algorithm to generate random numbers. And in this case, terminology gets a little tricky. I mean, I really have to call them "pseudorandom numbers," although I dislike the term because it implies that they're not good enough, or, you know, pseudorandom isn't as good as random. Well, in fact, really good pseudorandom numbers are so good that it can take an analysis of millions of them in order to determine that they're not random.

Leo: The problem in general with these is that they repeat after some period of time.

Steve: Well, really cheesy ones do.

Leo: Oh, not all of them do?

Steve: No, no, no. There are some...

Leo: Do they reseed it, is that how they change it, or...

Steve: Well, if you think about it, if we have, for example, 128 bits, and if the entire state of our random number generator were 128 bits, we know that 128 bits gives us a certain number of numbers. And if we ever hit on one of those again, we'd be stuck in a loop. But it turns out that really good random number generators use a huge pool of entropy, essentially, is one of the terms that they use, is they have a big pool of randomness. And the generator is constantly churning this big entropic pool. And so, for example, it might generate the number 5 five times in a row and then go off and never do it again.

Leo: Right.

Steve: So, I mean, really good pseudorandom number generators exist now. And, I mean, they're so good they can be used without concern. And what's cool about them is essentially you start them at a known state, and you give them a key which determines the sequence of randomness. Well, now, we've seen this before, and we talked about it very early on in the Security Now! podcast because RC4 was the random, sorry, the pseudorandom number generator used in the original WEP {Wired Equivalent Privacy) Wi-Fi encryption, which we know was so poorly implemented.

Leo: And it was that seed that was the problem.

Steve: Well, it was that the people who used it didn't appreciate the limitations. And one of the things that was really the Achilles heel is that the same number, that is, the same sequence, got used multiple times.

Leo: Like using a one-time pad more than once.

Steve: Exactly. I mean, that's exactly what it was. And so they created some technology that wouldn't have this happening all the time, but the number of total different sequences was not large enough that, if you listened for a while, you couldn't pick up some reuse. And as soon as you pick up reuse of the same sequence, you're in trouble.

Leo: Interesting.

Steve: Okay, so let's go back. We have RC4, which is a very nice pseudorandom number generator. What you need to do is you need to avoid what are now known as some weak keys that just don't initialize it very well. Because basically it starts out with an array, a 256-byte array that's filled with the values 0 through 255. The key is used to sort of pre-scramble that. And then, as you generate numbers with it, it continues to churn within that array additionally.

Well, it turns out that all you have to do is throw away the first maybe 256 bytes that it generates because those tend to not be very random. It sort of takes a while to get going. And again, using these tools correctly is certainly important. And, you know, the horrors of original Wi-Fi show us all the ways in which you can go wrong and make mistakes in not using these properly. But given that you had this, basically you could take a key and use RC4, throw away the first 256 bytes that it emits, then take the balance and simply use that as your pad. So, I mean, and it's really good encryption. Now you've got a pseudorandom sequence which is being used to spin our encryption around the wheel until we go the number of steps around, for example, clockwise of the pseudorandom number that's been emitted, and that's our symbol. And because the pseudorandom sequence generator is so random and so good, it is also incredibly difficult to decrypt. Now, we need to back off a little bit from impossible. True randomness that is mixed with a message cannot be decrypted. But because an algorithm was generated - I'm sorry, because an algorithm was used to generate the sequence, it's not absolutely truly random random random.

Leo: Right.

Steve: You know, it's really good random, but it's not - it is, okay, pseudorandom.

Leo: Right.

Steve: So there's that tiny weakness. But with really good pseudorandom number generators, encrypting a message with virtually state-of-the-art technology is just that simple. You use an algorithm to produce random numbers, and you spin around the wheel.

Now, I want to talk about one last thing here for this first, you know, into the technology of cryptography. And that's an operation which is performed all the time known as an Exclusive OR. In math, you know, there's like the basic functions OR and AND. For example, if you had two bits, and you run them through an AND function, you only get a 1-bit out when all of the inputs are true, or you're feeding 1 in. With an OR, you get a true value out, or a 1 out, if any one or more of the inputs is true. Then the OR function says, okay, I'm going to be true. An Exclusive OR is an interesting animal. It has two inputs. And the output is true if either input is true, but not both. So if you make a little chart for yourself where you say, okay, you've got two inputs that are going to be 0 and 0, so the output is 0. Now, the first input is 1, the second input is 0...

Leo: True.

Steve: ...and the output is true, exactly, 1. And the reverse. The first one is 0, the second one is 1, or true, then the output is true.

Leo: So it's only false if both are false.

Steve: Or if both are true.

Leo: Oh, it's false if both are true.

Steve: Yes, that's the key. So if the first input is 1 and the second input is 1, then the output is

0. So that's why it's called an "Exclusive OR." It's not...

Leo: Exclusive OR as opposed to a regular old OR.

Steve: Exactly. Well, what's very cool about this is, if you look at this chart, you'll realize that if you had, like, a signal coming in the first input that was going through the output. Now, let's say that the second input is 0. So we have our signal coming through. If the signal is 0, the output is 0. If the input signal is 1, the output is 1 because our other input is 0. So this is having no effect. But if our second input is a 1, look what happens. We put a 0 in, and because the other input is a 1, we get a 1 out. We put a 1 in, and because both of them are 1, we get a 0 out. In other words, this is inverting. It's sort of a conditional inverter. If our second input is 0, there's no inversion of the data. But if our second input is 1, anything we put in the input is flipped over. We put in a 1, we get out a 0. We put in a 0, we get out a 1. So that's what an ExOR operation does.

Leo: Got it.

Steve: Okay. So the reason that's cool is that it's reversible. That is to say, if you - say that you had your clear text, and you ran it through this ExOR, and it may or may not have inverted it. Okay, then you run it through the ExOR again. Well, if it had inverted it before, it will invert it now. And if it didn't before, it won't now. So the idea is it reverses the process. And that ends up being a key function in cryptography.

So now, instead of talking about our symbols, let's get away now from our little 28-symbol wheel and talk about bytes. Because bytes is the way everything works in, you know, everything that we're doing today. So we have an eight-bit byte of clear text. That could be, you know, the whole ASCII character, it could be a byte of computer data, any eight-bit pattern. We then ExOR that byte with an eight-bit pseudorandom value that came from our RC4 generator or whatever pseudorandom generator we're using. And that's going to produce this cipher text. And so we're done, basically. By just taking our clear text byte and ExORing all the bits with the pseudorandom bits from our generator, we now have encrypted our byte. And literally, that's all that is being done today.

Leo: And it's reversible.

Steve: And that's the beauty because, if you then take that and you send it somewhere, you store it somewhere, you do whatever you want to with it, doesn't matter because, I mean, it is, at this point, after running it through that ExOR operation, ExORing it, and that is to say, mixing it with a pseudorandom value, it is now ciphered. That's all there is to it. You then, if you want to decipher it or decrypt it, someone wants to read your message, all they do is they do need to have the key for the pseudorandom sequence generator so that they're going to be spitting out the same random bytes. But they take your byte, ExOR it with the byte which will have been the same one you used, and because this ExOR process is reversible, out drops your original byte.

Leo: Isn't that slick. I've always liked ExOR. Always been one of my favorites. It's so elegant.

Steve: Well, it's just - it's clean. And what I love about this is this is so simple. I mean, what I'm describing is exactly how many systems work today. And as I promised you, Leo, that's all

there is to it. I mean, everyone can understand this.

Leo: Yeah.

Steve: This is not, I mean, on some level I guess it's rocket science, but, you know - and in fact, the details of the pseudorandom sequence generator that's spitting out these really good random bytes, that's pretty hairy, you know, cryptomath stuff. But if you have a stream of pseudorandom bytes, you simply ExOR them with your data, and you get something out nothing in any reasonable time can crack.

Leo: It's really remarkable. Now, I wish we had illustrations on this one because this is a tough one to do in audio. I mean, I am impressed that you were able to explain this. And I followed it, but it is tough.

Steve: Well, I want to try to do this because we're doing an audio podcast...

Leo: Right.

Steve: ...and draw the examples. I think that falling back to the decoder ring in order to sort of build up some vocabulary, and then moving into bytes, I would imagine we've probably held everyone here.

Leo: Yeah.

Steve: And so essentially what we've just described is, I mean, absolutely symmetric key encryption as it is used today. We have a key that must be kept private because that key allows someone to generate the same stream of pseudorandom bytes which are then ExORed against the cipher text in order to recreate the original plain text. So, and that's one of the keys.

Now, you know, you mentioned early on, of course, the problem with the one-time pad was that the person on the receiving end had to somehow get that ahead of time. Well, we've reduced the problem's size, but we have not at all reduced the nature of the problem because the key still must be kept private. In the same way that the suitcase with the one-time pad had to be kept out of enemy hands, even though we now have a key that lets us easily generate as much data as we want to, which is very cool, the secrecy of that key is every bit as important. And now we're going to - next time we're going to talk about...

Leo: Shh, don't tell them how we're going to solve this problem.

Steve: It turns out there are very cool technologies that solve that so that it's possible to establish a communication with another party where somebody eavesdropping can't gain the information they need to decode the conversation.

Leo: And that was the real revolution in cryptography really.

Steve: It's a very cool thing. So here I just want to make sure people get it, that this key that generates the random numbers - now, notice not the key itself, or not the key alone, but also all of the machinery for generating the random numbers, that has to be known, too. So somebody who was going to decrypt it basically needs to recreate the same stream of pseudorandom data to ExOR against the cipher text to create the clear text. So they have to know what algorithm you're using, the details of that, and the key. So more than just the key is necessary. But the problem is, all that stuff has to be kept secret. And we know that, you know, there is some security in obscurity, but it's not nearly as good as any security technology where you can just be completely open and say, here's what we're doing, give it your best shot.

Leo: Wouldn't it be neat if we could just give our keys out and still have security. Hmm.

Steve: And we'll be talking about how that works in the coming weeks.

Leo: Now, we won't do it next week because next week is a question-and-answer week, our 32nd episode. Every fourth episode Steve sits down and answers the mailbag, and there's a bunch of great questions for you, I know.

Steve: And now, thanks to this updated form on the page, we'll have a better sense for who and where people are, and I'll be able to answer people's questions, and we'll go from there.

Leo: Don't forget to go to his website, GRC.com/securitynow.htm, for transcripts and 16KB versions of the show, all his show notes and links and so forth. And of course, while you're there, check out SpinRite, which is the ultimate disk recovery and maintenance utility. I use it, and I encourage everybody to get a copy of SpinRite, Steve Gibson's incredible program.

Steve: Well, and now we've got SpinRite.info...

Leo: Makes it easy.

Steve: ...for anyone who wants to see some listener feedback, yup.

Leo: Just go to SpinRite.info. I can't believe you, Steve. This is a revolution. A revolution. Have you ever had anything besides GRC.com?

Steve: Actually I own SpinRite.com. And that brings you to my site also. And I've got various other little domains that I use here and there. But, you know, GRC.com, of course, is the flagship.

Leo: That's the one. That's the one. Well, we thank our flagship, AOL Radio. They broadcast us on their podcast channel and of course provide us the bandwidth so we can get this to you via podcasting. That's AOL.com/podcasting for more information. They've got some great programs on there.