Symmetric Block Ciphers

Description: Leo and Steve answer last week's Puzzler/BrainTeaser which explored the idea of using two private one-time pad "keys," like two padlocks, to securely convey a message between two parties, neither of whom would have the other's key. They then continue their ongoing tour of fundamental crypto technology by describing the operation of Symmetric Block Ciphers.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-033.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-033-lq.mp3

Leo Laporte: This is Security Now! with Steve Gibson, Episode 33 for March 30, 2006: Symmetric Block Ciphers. Hello, Steve Gibson.

Steve Gibson: Hey, Leo. Great to be back with you.

Leo: Are you swamped with answers to your stumper from last week?

Steve: Oh. Oh. I thought I was getting a lot of postings on the Security Now! page before. I mean, it's - well, the really cool thing is that people have had a lot of fun with this. I mean, I've received things that look like Ph.D. theses with pages covered in mathematical expressions with XORs and proofs of why, you know, it does work that you can communicate, but why it's not secure. We had a housewife who wrote, said she's out walking her dog, listening to our podcasts, and either we're doing too good a job teaching or these puzzles are not hard enough, and then she completely explained why this is not secure. I mean, and people have had a...

Leo: And she got it right.

Steve: Oh, yeah. And, in fact, many people pointed out that one of the problems is that this is an abuse of the idea of a one-time pad. Essentially, it's been used as a two-time pad because people are using that pad twice.

Leo: Well, let me re-pose the stumper so people know what we're talking about, in case they missed Episode 32. And then you can explain why it doesn't work.

Steve: In fact, if they missed Episode 32 they ought to go back and listen to the beginning of it because - you know, like, stop now, go listen to the beginning of it, and you can have a puzzle all over again.

Leo: Yes, that's true. Then you won't know the answer. That's fun. In fact, I think we should do this more. I think people really like it.

So the premise was we were talking about one-time pads a couple of episodes ago and how they really work. But the issue was you have to get the pad to the recipient, so that he - or get a copy to the recipient. And there's a vulnerability because, in that transmission of the one-time pad, somebody could get it. So somebody suggested, well, here's a clever idea. Why don't you double encrypt? So you have two individuals with independent one-time pads. Individual A encrypts using his one-time pad, sends it to Individual B, who encrypts it using his one-time pad, sends it back to Individual A - am I right so far?

Steve: Yep.

Leo: And then Individual A...

Steve: Decrypts.

Leo: ...decrypts his portion, sending it - now, it's still encrypted - sending it back to Individual B, who then decrypts it with his portion.

Steve: Right.

Leo: And at no point was clear text sent back and forth, nor was the one-time pad exchanged. And you didn't even say that there was something wrong with this. You just said, would this work?

Steve: Well, because that was the question posed to me by a listener after our previous episode. So it's like, okay, let's let this open to question, you know, does it work. And several people commented that yes, it works, that is, it does always transmit an encrypted message. And because of the nature of the simple XOR encryption or adding and subtracting, you are able to do these out of order.

Now, normally if you encrypted something and then you encrypted it again, you would not be able to decrypt it with the first key. That is, you always have to back yourself out in the same sequence, like in the reverse order of sequence of decryptions from encryptions, in order to completely reverse the process. But because this is a simple XOR, or an adding and subtracting, like going back to our secret decoder ring, those operations are commutative, and you are able to do them out of order. So, you know, Person A encrypts, then B encrypts, then A decrypts, which still leaves B's encryption, which is then the final process that decrypts. So in this simple case it does work to do the encryption and decryption out of order.

Leo: So the flaw isn't in that respect.

Steve: Right.

Leo: It does work.

Steve: Right. Yeah. Here's the problem. We have to presume that there's an eavesdropper. I mean, that's the whole point of encryption anyway is that we want to send an encrypted messages that, if somebody were to intercept it, it wouldn't help them. So an eavesdropper sees the message going from Person A to B, which has been encrypted with Person A's one-time pad. They also then see the message going back from B to A after B has encrypted it. Well, there's the problem. Because if you see the message before B encrypts it with his and after, the encryption is so simple that essentially you just - so you XOR those two messages, or you subtract one from the other. And that gives you B's one-time pad.

Leo: So as our dog walker pointed out, your flaw here is that you're using a one-time pad twice.

Steve: Well, yeah, and you're able to see the information. So by seeing the first and the second exchange, you're able to get the one-time pad used by the second person. Then, when the first person removes their encryption, the message is now only encrypted with the second person's one-time pad, which you have. So now you simply decrypt it with that pad, and you've decrypted the message by seeing it in transit. So this is clearly, I mean, it works, but it's anything but secure.

Leo: You know, I think we have some very brilliant, mathematically-inclined listeners, because I wouldn't have gotten that. That's very impressive.

Steve: People had a lot of fun with it. In fact, a poster named Steve in Chicago also pointed out - and I thought this was really cool - that the original concept that introduced this, the idea of a double lock-box where, instead of using encryption - and this was what the original poster suggested - instead of using encryption, you'd put a lock on a box, send it to a friend, like, okay, I lock a box, send it to you, Leo. You add your lock to it, send it back to me. I take my lock...

Leo: I can't open the box, but I can put another lock on it.

Steve: Exactly. Then you send it back to me, I remove my lock, leaving it now only locked with yours, which I send to you, and you unlock.

Leo: So it's locked the whole way, and I don't need your key, and you don't need my key.

Steve: Yeah. And so anyway, a listener pointed out that is subject to a man-in-the-middle attack, which I thought was really cool. Someone intercepts my sending it to you and pretends to be you. And that's the problem with man-in-the-middle attacks is a lack of authentication. I mean, that is the thing that is necessary in order to prevent this type of attack.

Leo: So he puts his lock on it and pretends to be me.

Steve: He puts his lock on, yup, sends it back to me. I remove mine, send it back to him, he removes his. Now he opens it, takes a look at the inside, changes it, does whatever evil thing he wants to to it. Now he pretends to be me, sending it on to you. So he basically does the same three-way handshake from him in the middle to you, and you don't know you didn't get the contents from me.

Leo: Right.

Steve: So I thought it was just another very clever, real world analogy to this kind of stuff.

Leo: That's exactly how a man-in-the-middle attack works. Interesting.

Steve: Yeah.

Leo: So we're going to continue on now. We have crypto to talk about. We've done 101 and 102.

Steve: Oh, we've got crypto for weeks. There's so much cool stuff in crypto. And what I think people are going to find is it's surprisingly straightforward. I mean, it seems like rocket science, you know, mumbo-jumbo, confusing stuff. But essentially there are a series of simple building blocks which each block is clear and easy to understand. And then, once we have those, we're going to look at all the different ways they can be put together to do real work. And it's just so cool.

Leo: I want to add one little note. A couple of people sent me - and I'm sure you got the same emails - recommendations for books along this line, two of which I've read and really can agree with are excellent: Simon Singh's "The Code Book," which is a very good description of how crypto works and crypto through the ages; and then, if you really want to know everything about how it's been used, David Kahn's classic "[The] Codebreakers" is quite amazing.

Steve: Well, those are good sort of literature-oriented books. My bible is Bruce Schneier's "Applied Cryptography."

Leo: Yeah, Bruce is the king, isn't he, yeah.

Steve: Oh, he just is. And "Applied Cryptography" is, I don't know what, like, fourth edition or fifth or sixth? I mean, I've got it in hardback and softback. I love it. Basically it's the bible of this stuff. I mean, it's heavy-duty technology. But if someone really wanted to understand without any simplification, Bruce lays it out. And it's a fantastic reference.

Leo: Tell you what, I'll put links to all three books in the show notes, and you can look at the Amazon reviews and decide which one is right for you. I'm not ready for "Codebreakers" - I mean, for Bruce's book.

Steve: "Applied Cryptography," yeah.

Leo: But "[The] Codebreakers" and "The Code Book" were right for me. Shall we - what are we going to talk about this week?

Steve: Okay. This week is the second type of symmetric cryptography, or symmetric ciphers, or symmetric encryption/decryption, known as "block ciphers." What we covered last week are known as "stream ciphers" because - wait, no, it was week before last, I'm sorry, before our Q&A episode, two weeks ago. Those were stream ciphers, you know, taking a look at RC4, which of course is the famous one that was used in WEP - which, because it was not used correctly, even though RC4 itself is a very good source for encryption, it was not applied correctly. The most important thing that the people who are applying this technology have to be aware of is using them correctly. You can have a really strong tool which you don't use correctly and get yourself in trouble. So...

Leo: So you've digressed a little bit. But what's the difference between a block cipher and a stream cipher?

Steve: Well, that's where we're going to go right now.

Leo: Okay.

Steve: The idea was, you'll remember we had basically a stream of bytes, and if we had a one-time pad, then we had a true source of random numbers or symbols that we were mixing with our message in order to produce a truly random result. So that was like the pad represented a stream of randomness. If we didn't have a one-time pad, we could instead use a cryptographically strong pseudorandom number generator. So this again, just every time you ask this generator for a new random number, it hands you one, the idea being that a very good mathematical algorithm is producing these so that no analysis of them can find a pattern. And we know how to do that now. Our modern cryptography - in fact, RC4 for its simplicity, and it's a very simple random number generator, which is why it was originally adopted by the WEP guys in the beginning was they wanted to be able to run this on very inexpensive, low-power hardware and low-power machines. So it's very good. It turns out you have to look at millions and millions of random numbers produced by it to even be able to tell it's not ping-pong ball random. I mean, it's not truly, absolutely random. It's that good.

So the idea is you have a pseudorandom number generator that just spits out this stream of random bits, which you mix in with your message, turning your message into cipher text - into, essentially, random bits. And then at the other end, of course, somebody with the same pseudorandom number generator and the same key, because these are keyed so that the specific sequence of pseudorandom bytes or bits that it generates is dependent upon the key. So they generate the same stream of pseudorandomness. They unmix it with the cipher text, returning it to plain text. So that's stream ciphers.

There's an entirely different way, which has actually become recently the dominant approach for cryptography for, like, bulk encryption. That's what SSL and OpenVPN, it's what the new WPA encryption uses. Basically, stream ciphers are kind of old school. They still have a place, and they were a nice place for us to start. But now we're going to talk about what are called "block ciphers."

We'll start by going back to our decoder ring. Now, with the decoder ring, we set our wheels on the concentric circles of the ring to some position and basically just had a simple transposition, that is, from any character you'd move a certain number over to the other one. And that was encryption. And to decrypt, you just went in the opposite direction. Now imagine an entirely

different approach. You have a table of every possible symbol in the alphabet. So you just write down A through Z down one column. Now on the other side you just fill in the second column in a completely random sequence. So, for example, somewhere on the second column you put A. Somewhere else you put B. Somewhere else you put C. Somewhere else you put D. So basically you create a mapping between your clear text or plain text data and your cipher text. But in this case, there is no relationship between the columns. If we compare this to the previous approach, one column would just be shifted down or transposed from the other, making it very simple to, like, you know, Caesar's generals were able to decode this, apparently. But here we just create an arbitrary mapping between our input symbols and our output symbols, so that the result is completely scrambled. Now, that's a perfect example of a very simple block cipher.

Leo: Okay.

Steve: So again, obviously the process of encrypting this is you find the symbol on the left, and you simply do a table lookup. You just look up the corresponding arbitrarily assigned symbol. But it's statically assigned. That is, every time you ask for the same symbol on the left, you're going to get the same symbol on the right. And so you produce your message. Now we have the same problem we've had before, that is, to decrypt you need to get - the person who's going to decode it has to have the same, basically a copy of that table.

Okay. So we go to the first step, which is, rather than creating - well, okay. We have to talk about the weaknesses of this first. Obviously this is not a very strong cipher, if you know anything about the language, because as we talked about before, even with the transposition cipher, if you did an analysis of the frequency of occurrences of these symbols, and you know, for example, that the message was probably going to be enciphered English text, you just look at it, and you say, oh, look at the - based on the frequency analysis, this symbol, you know, which might be Q, for example, is obviously E in English because it's occurring so often. So it would be very simple to crack this simple mapping cipher, this very simple...

Leo: Substitution cipher.

Steve: ...substitution cipher, exactly, or block cipher, by just doing an analysis of the frequency. Now, so we have the idea of a random assignment. Well, now we have algorithms which are becoming very popular and work very well, where a key automatically creates this mapping. So instead of sitting down and just, like, randomly assigned these letters, we can use an algorithm to map from the plain text to the clear text. And that algorithm is driven by a key with a certain number of bits. And that, of course, is where we get our strength.

Now, in order to solve this problem of the frequency analysis of the result being so simple, two things are done. The first is we don't use a single symbol block. That is, in this first example, the so-called block length of our cipher was one symbol, one character, maybe eight bits. Or actually, you know, if we were just using an alphabet, we could get away with 32 bits to hold the whole alphabet. I'm sorry, five bits, because that's 32 possibilities, and that would be enough for A through Z, space and dot and so forth. So the way we solve the frequency analysis problem is we encode many more symbols at a time. Contemporary block ciphers are 64 bits long. So that's, for example, in terms of eight-bit bytes, that's obviously eight characters. So when you take eight characters or eight symbols in a block, what you're doing - and this is what's fascinating about these algorithms is, I mean, 64 bits is a huge number of possibilities. We know that 32 bits is 4 gig. So that means that 64 bits will be 16 billion billion possible combinations.

Leo: That seems like enough.

Steve: Yes. And here's what's so cool, is any 64-bit combination you put into one of these block enciphering algorithms produces exactly one other, that is, one matching 64-bit combination. So, I mean, this is phenomenally strong. You put in - I'll say it again. You put in eight bytes at a time, or 64 bits, and a completely different, sort of like mapping to a different 64 bits, comes out. And what's so cool is, if you just changed one bit on the input, just changed one of those 64 bits from a 0 to a 1 or a 1 to a 0, what comes out is completely different. It's not like one bit over on the other side changes. Everything changes. It's just a completely different mapping. And that's what...

Leo: That's cool.

Steve: Oh, it's such cool technology. And that's what makes this so strong is, even if you tried to do an analysis of the frequency, you would just, you know, eight characters in English don't occur with specific probabilities, you're not going to know where word boundaries are. Even if you had, like, lots of E's and T's and S's, they're going to be watered down by lesser occurring characters in their neighborhood, so that there's just no way to analyze the frequency strength of what's coming out and figure out what went in. Basically...

Leo: That explains a lot to me because I'll look at an encrypted PGP message, and the length doesn't even seem to correspond with the length of the clear text message.

Steve: Well, and in fact it won't because everything has to be done in blocks. So the size of the message will be rounded up to at least the next even block size in order to finish off the message. You have, like, three characters after you're done chopping it up into sets of eight. And then you just have to pad it with an additional five in order for it to be an even number of blocks. So that's one thing that's different about the use of a block cipher versus a stream cipher. As we've seen, a stream cipher can be exactly as long as the message because you're just done when you're done.

Leo: And even that gives you some information that could be useful, is the length of the message.

Steve: Sure.

Leo: Yeah.

Steve: Sure. So what's so cool about these algorithms is it's a virtual impossible to figure out mapping between a huge number of possible input combinations, you know, 16 billion billion, and the same number of outputs. And of course this is reversible, that is, these algorithms, you have an encryption algorithm that goes one way, driven by a key, and the decrypter is typically a different algorithm because it's done so much to this data inside. But still you use the same key. Thus it's a symmetric key cipher, that is, the same key drives a different algorithm in order to reverse this process. So it takes any one of this huge number of 64-bit inputs and unwraps it, unwinds it, and returns it to what was originally put in.

Now, the last thing that is done in order to further strengthen this is there would still be some

weakness if you had these eight-bit blocks all standing by themselves. I mean, the problem has been made extremely difficult. But, you know, it's like, well, okay, maybe if you did analyze this you could, you know, if you had enough cipher text and enough time you could still do a frequency domain analysis of the result in order to see what was going on. So what they do is something called "cipher block chaining," or CBC for short. And essentially what they do is they take the prior block and XOR that - here we go with our favorite XOR operation again. You XOR it with the data for the next block before encrypting it. What that does is that creates an interblock dependency so that no longer does each eight-bit block stand alone, which would potentially render it vulnerable to some sort of cryptographic analysis. Now you have to know what the one before was in order to XOR that with the one that follows. So it's a reversible process; but basically it means that, if anything changes in the message, the entire message is destroyed. Whereas, if we had eight-byte blocks standing alone, something could change there, and the rest of the message would not be hurt. So this strengthens the enciphering of the message, and it makes it, you know, way harder for someone to cryptanalyze because they don't know what came before, so these individuals blocks are not standing alone.

Now, let's look a little bit at the history of this. The most famous block cipher is DES, the Data Encryption Standard that was adopted by the U.S. government and has been used widely for a long time. DES is a very good and strong block cipher, but it suffers from a key length which is too short. The DES algorithm uses only a 56-bit key. Now, once upon a time that was a lot. But, you know, over the last decade computers have gotten so much faster that 56 bits is no longer enough. And if somebody wanted to, if a government, like government-level funding produced a DES-cracking machine hardware - and, by the way, several have been produced - then it's possible in a relatively short time to brute-force attack the DES block cipher, just because 56 bits of key doesn't give you enough possible combinations. You can just try them all, and you're going to be able to get there sooner or later.

So what was done, as sort of one of the many improvements in general for block ciphering, something called "triple DES," or 3DES, was created. 3DES uses two 56-bit keys and three DES operations. So it takes the same basic 56-bit encryption algorithm, but it uses it three times with two different keys. Well, these two different keys, essentially the two 56-bit keys form an effective single 112-bit key. And 112 bits is enough to give you, in today's world and for the foreseeable future, very strong encryption. So many people have gone to 3DES as a nice way of strengthening DES and for bringing it up to current standards.

Leo: So they wanted to preserve the DES standard, but they wanted to make it more. So the strength of DES is that it's very easy to implement.

Steve: Yeah, exactly. DES was chosen because it was fast and easy to implement. And it has been pounded on so much. See, it's like any brand new cipher might have a problem. This is the same thing we run into with security. It's like any brand new software, any brand new operating system, any brand new anything in security, you know, a brand new lock might have some way that the lock makers didn't think of for it to be cracked or hacked. You just don't know. It's really - it's only history and experience that allows us to prove the security of these technologies. So...

Leo: So it's sometimes better to go with a tried-and-true method, even if it has some flaws. Just fix the flaw and continue to use it.

Steve: Oh, absolutely. In fact, you know, every so often you'll see some website that's bragging about their own hyper-triple-duty, super-triple-scoop encryption. And it's like, what are you people thinking? You know, this problem has been solved already.

Leo: Right.

Steve: And that's what I love about this is there is a new standard. It's known as AES, or the Advanced Encryption Standard. It's actually a cipher called "Rijndael." Rijndael was chosen from among a long list of ciphers in an international competition. So all these cryptographers who had their own favorite ciphers - and there are many block ciphers. You know, people have probably heard of Blowfish.

Leo: Right.

Steve: And actually Blowfish was invented by Bruce Schneier, and he put it in the public domain. I mean, it's a very good, very strong block cipher. There's something called CAST, Crypton, DEAL, DES of course we talked about, DFC, E2, FROG, HPC. IDEA is the International Data Encryption Algorithm. And in fact, that's the bulk encryption used by PGP.

Leo: By the commercial PGP. It's licensed, so it can't be used by the open source one, yeah.

Steve: Exactly. LOKI97, Magenta, MARS. Then there's RC2, which is a block cipher, RC5 and RC6. And actually RC6 was one of the ones in the running for the AES. And Safer and Serpent and 2fish, I mean, so the point is there's a whole pot of really good block ciphers. And what WPA uses, it can use, as we've talked about before, when it's running on older hardware it can use what users see as TKIP, the Temporal Key Integrity Protocol. That's actually still RC4 running in the background. But they fixed the way it's being used so it's completely secure. But the alternative, and you'll see this sometimes in your user interface on your routers and maybe, you know, XP or something, depending upon what your Wi-Fi stuff is, you may see AES as an alternative cipher. It's more compute intensive, so you will technically get a slower connection. And since it's sort of overkill, I don't use it, and I don't even recommend it. But it is there, and it's a block cipher. It was like the next generation sort of beyond DES as the formal, world-wide standard. And boy, I mean, it has been pounded on extensively.

Leo: So we know we shouldn't use DES, but 3DES is fine.

Steve: Right.

Leo: Blowfish is fine, IDEA. Are they all roughly equally strong?

Steve: Well, there are various differences between them. They're generally all regarded as strong. Really, given a strong block cipher algorithm, the idea is that the only thing that should matter are the block length and the key length, that is, the block length, a longer block length is generally better because it defeats any kind of frequency analysis by just muddying up any interrelationships and frequencies of the input bytes. It just mixes them up so much you can't see much statistical value in what comes out. Basically it's turning what you put in into just random bits. You know, once again, that's the idea. And in fact, it's really interesting, a good measure of the quality of encryption is the compressibility of the output. Your output should not be compressible because...

Leo: The more random it is, the less you can compress it.

Steve: Exactly. And so you want encryption to basically create a completely random output that isn't compressible because there's no patterns for a compressor to find in order to leverage in order to squeeze down the result. So a modern block cipher like the AES-chosen Rijndael, for example, it allows you to use a key length of 128, I mean, it doesn't even have anything shorter than 128. It's like, okay, we're not doing less than 128 these days. So you can run it at 128, 192, or 256.

Leo: Wow.

Steve: And so 256, I mean, that is just so beyond reason. But it works, and so why not?

Leo: Is it highly compute intensive? I mean, is it so compute intensive that you wouldn't want to use it? Or nowadays should we just go ahead and use it?

Steve: I guess it depends upon your application. For example, our SSL connections are currently being encrypted with 128-bit security. And that's really sufficient because they're not static keys.

Leo: Right.

Steve: That is, every time I create a connection to a server, the handshaking that goes back and forth - which is so cool, and we'll be talking about it in a couple weeks - both ends agree on a random 128-bit key just for that connection. And then it's scrapped and thrown away, and another one is chosen for the next connection. So the lifetime of these keys relative to their strength really renders them, I mean, 128 bits is really strong. So, yeah, a longer key length will generally require more processing, more hardware, more software, whatever. But our machines are so fast now that, if you really wanted to use them, it's why, you know, why not? And in fact, that's what I've done with our OpenVPN configuration is I'm using 256-bit symmetric block encryption.

Now, it's worth mentioning that these ciphers, these block ciphers are extremely fast relative to what we're going to be talking about next week, which are the so-called "public key" or "asymmetric" ciphers, where you use one key to encrypt and a different matching key to decrypt. When these algorithms are implemented in hardware on the order of 1,000 times faster, that is, the block ciphers are 1,000 times faster than the public key systems. So these are - that's why they're also called "bulk encryption ciphers." For example, it's the bulk - IDEA is the chosen bulk encryption, as you mentioned, used for the commercial version of PGP because they're very, very fast. You wouldn't want to transfer lots of data using asymmetric ciphers that we'll be talking about next week because they're on the order of a thousand times slower. And so we'll talk about how these all work together, as I was talking about with building blocks, in order to give us a really cool set of possible functions.

Leo: So there isn't any blanket recommendation that you would make for a particular technology. It really depends what you're doing.

Steve: Well, it does. And, for example, a perfect example was Schneier's Blowfish, that is a

very strong block cipher that he put in the public domain. It's available to everyone. Only recently did the patents on RSA's ciphers expire. So, for example, you couldn't legally use RC4 or RC2 or RC5, which they had patents on.

Leo: Right.

Steve: You couldn't use them without licensing them from them. Only now can you. And in fact...

Leo: And it's the same issue now with IDEA, which is patent encumbered, so...

Steve: Yes, exactly. So...

Leo: But in time you'll be able to use that, as well.

Steve: Yeah. And in fact, of course one of the requirements for any cipher that was adopted to be the AES, the Advanced Encryption Standard, it had to be released into the public domain. And this RC6 cipher that RSA put up for adoption, it is not publicly available. And they said, if our cipher, RC6, is adopted, then we will release it into the public domain, but otherwise not. So certainly one of the issues for choosing a block cipher, or any cipher, stream or block cipher, is what are the licensing constraints, what are the, for example, the key lengths, the block size, and so forth. So they have personalities and characteristics that, when you are sitting down to choose what you want to do, you know, those things would come into play.

Leo: I always just take the default. I presume that whoever wrote G or PGP is picking the right one for me.

Steve: Right. And I think that's probably a safe thing to do.

Leo: And so that really gets back to my fundamental question, which is they're all secure, they're all relatively secure; right?

Steve: Well, generally, none of these have been cracked. You know, you'll see people posting things saying that a cipher has been cracked. Well, what that would really mean is that you could trivially determine the key, or you could trivially reverse the encryption process. What's normally done is that, and what they mean when they say "cracked" is that, once upon a time, through a tremendous effort, one key was determined in some sort of a challenge. And, for example, companies like RSA create public challenges, saying we'll offer prizes, we'll offer recognition to academic establishments and anyone who wants to try to come up with the key. So they'll say, here's the cipher we're using, here's the plain text, here's the cipher text. What is the key that we used in order to produce this? And it's one of the ways that these ciphers are strengthened is just by giving some incentive for people to pound on them and see if there's some way to break them or crack them.

None of these have been cracked, although "cracked" meaning that basically it's rendered worthless. What has happened is that the shorter key lengths have been brute-force attacked. And that's the idea. You want an algorithm where there is no choice other than to try every single key. And then all you need to do is have a bit length on your key that is so long that

every single key becomes virtually impractical in the lifetime of the universe. I mean, we're at, what was it, 10 to the 19 years.

Leo: Right. You're right about people coming up with kind of crazy schemes. In his book "Cryptonomicon," Neal Stephenson claims to come up with a very strong encryption technique involving a pack of cards. And the point is, yeah, there are lots of different algorithms. But if you stick to these half-dozen well-known, strong algorithms, chances are they are strong because so many people are attacking them and trying to break them.

Steve: Well, and anytime you wander off the trail and try something cute, you really open yourself to vulnerability. I mean...

Leo: As the engineers who designed WEP learned, to their dismay.

Steve: Yeah. I mean, there just isn't a reason not to use these. I mean, it's like this problem is solved. You know, it's like no one needs to do new crypto because really, really strong algorithms with long bit lengths which have been well proven are in the public domain. They're free to use. Why not use them?

Leo: Always a pleasure, Steve. This is a fascinating subject. And I think we could do many, many more episodes on crypto. I think people are very interested in it. It's something that both captures the kind of imagination of the layman and also has some real practical import for any of us who use digital technology.

Steve: Oh, well, in a digital world, Leo, we are dependent upon the privacy and the security of the data we store and the messages we send. And more and more, as people are, for example, configuring Wi-Fi, they're checking their SSL certificates to make sure that the site they're connecting to is what they think it is, you know, these cryptographic concepts are entering and impinging on people's lives. So it's important. And in the coming weeks we're going to move through this exactly as we have been. And by the time we're done, people who've been following along, they are going to get all this stuff. They're going to, I mean, it's what's so cool about is it's not hard to understand. Here we've just explained, in a very useful fashion, symmetric block ciphers. Everyone who's been listening gets it now. And then we're going to talk about how...

Leo: To one degree or another.

Steve: ...how you use these. Well, I mean, no one's writing one of these algorithms, but that's the point. That's not necessary.

Leo: It's not necessary, right.

Steve: Yup.

Leo: Right, we don't need to. In fact, it also inspires us to, as you said, check and make sure and upgrade. Because of this, I went out and I looked at my - I've always used some

form of digital signing on my emails, and I upgraded it to the latest version of GPG, the GNU Privacy Guard, because I want to make sure I have the latest version. And I sign all my emails digitally, and encrypt many of them if people give me their key. So I think it's a nice thing to have available to you.

Steve: Yeah.

Leo: Hey, thank you, Steve. Next week what do we want to talk about?

Steve: Next week is asymmetric ciphers, so-called "public key cryptography."

Leo: That's what I'm using, of course, yeah.

Steve: Well, actually you're using symmetric. You don't know it, but you are.

Leo: Oh.

Steve: Anyone who is using public key is also using...

Leo: Both.

Steve: ...private key, or symmetric, for reasons we will cover next week.

Leo: Ah. All will become clear. And Steve, we've got to do more stumpers. People obviously like them.

Steve: Well, when I can think of something, or when someone presents something. I agree, it was a...

Leo: Send us your stumpers.

Steve: It was a fun thing.

Leo: Stump Steve, and we'll offer it to the group as a whole. Of course you can get more information on all of these subjects at Steve's website, GRC.com/securitynow.htm. That's also the home of SpinRite, Steve's fantastic disk recovery and maintenance utility, a must-have for every user. Anybody who's got a PC should have SpinRite. And you can find out more about it at SpinRite.info. And if you're looking for transcripts or 16KB versions of this show or show notes, again, GRC.com/securitynow.htm.

We thank our good friends at America Online for broadcasting this show on their podcast

channel on AOL Radio. And you can find out more about that at AOL.com/podcasting. And we'll see you next week on Security Now!.

Steve: Looking forward to it, Leo.

Copyright (c) 2006 by Steve Gibson and Leo Laporte. SOME RIGHTS RESERVED

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