ECC - Elliptic Curve Cryptography

Description: After catching up with the week's most important security news, Steve and Leo wind up their propeller-cap beanies right to the breaking point of their springs in order to obtain enough lift to examine and explore the operation of ECC - Elliptic Curve Cryptography - the next-generation public key cryptography technology.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-374.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-374-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson calls this episode a "propeller-head" episode. Get your thinking cap and your beanie on because we're going to talk about how something called "elliptic curve crypto" works and why you might start seeing it on many devices. That's next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 374, recorded October 17th, 2012: Elliptic Curve Cryptography.

It's time for Security Now!, the show that protects you Now!, with this guy right here Now!, the Security Now! host and Explainer in Chief, Mr. Steve Gibson. Hi, Steve.

Steve Gibson: Too much coffee for you, Leo.

Leo: Now! Triple tall latte. So I decided - you drink the big cups; right? The venti quinti, and you put a lot of shots in it. But I realized I don't want all that milk, so I just put a lot of shots in a small cup.

Steve: It was so funny, yesterday Starbucks, my Starbucks was demoing their new machine, which just came out, the...

Leo: The Clover. Did you get your Clover?

Steve: Oh, no, no. It's a little - it's a consumer pod-based thing.

Leo: Oh, that's right, yeah.

Steve: The Verismo or something like that. It's trying to sound verasic [sic] or something, I don't know. Anyway, and so they were like, I was walking around, and they said, would you like to try a latte from our new Verismo? And I said okay. And they said, "But it's not six shots." It's like, oh, okay, well, maybe we'll use three of them lined up or something. It was a little disturbing. They have a little pod. It's a pod. We've seen pods before.

Leo: I'm not impressed.

Steve: Pod of espresso. They had a pod of milk. And I was looking at it...

Leo: No.

Steve: ...thinking, okay, wait a minute, how did a latte come out of this little pod of milk? Then, because I got my curiosity up, I did some research, and it's powdered, finely powdered milk.

Leo: No, no, no, no, no.

Steve: And it's like, eh, don't think so.

Leo: No, no, no.

Steve: And it's $200 for this little pod-based system. And it's what they're pushing now. They have a bigger brother that's got three times the tank size. Then it tells you when the water's getting low or the used pod storage tank is getting full and so forth.

Leo: I am extremely happy with my Breville Dual Boiler. I have to say, I am getting the best coffee out of that. And the Vario, the Baratza Vario Grinder. I mean, I spent a lot more than 200 bucks on it. But I am - it's the best coffee. And now when I drink Starbucks or even Peet's, it's like, wow, I miss my coffee.

Steve: Yup. That's the way to be.

Leo: It's nice when you can get some really good espresso out of your own home machine. But that's not what we're here for today.

Steve: No, indeed, it is not.

Leo: What are we here for? I see something that worries me.

Steve: Yeah, it should.

Leo: Elliptical curve cryptography? You're kidding me; right?

Steve: We've been doing a bunch of softball episodes for a while. They've been interesting, but they haven't been very challenging.

Leo: Steve's going to throw the high heat today, kids.

Steve: So, yes, we need to wind up our propeller beanies' springs right to the breaking point in order to get enough lift for this. This is something important that we've referred to from time to time. We have never done a, okay, Explainer in Chief mode, how does this work? And that is Elliptic Curve Cryptography, ECC. And we have an acronym collision, of course, because ECC is near and dear to my heart as Error Correction Code, that I've been living with ever since the beginning of SpinRite. This is a different ECC. This is an alternative technology for public key crypto as opposed to private key or symmetric crypto, which we've talked about extensively.

We've also talked about the typical RSA-style crypto, the famous, you take two prime numbers and you multiply them in a modulus field, in a finite field, and the point being that we're relying on the difficulty, as far as we know, of factoring large numbers into primes, that is, the difficulty of determining prime factors. We're relying on that hard problem for all of the protection that we get from existing sort of standard RSA-style public key crypto. And the problems with that type of crypto are several that we've talked about. One is, primarily, it's slow. Which is to say it takes a huge amount of work for a processor to perform the operations needed because the actual protection is relatively weak. That's why we need, like, 2048-bit keys in order to get state-of-the-art protection, so if the protection level is weak, so we compensate for that by making the keys long.

Elliptic curve cryptography has been around already for a couple decades. It's not like it's a brand new invention. But it's coming into vogue because we're becoming more interested in performance and in lower power applications. There are some things coming up I can't talk about that I needed to lay some groundwork for. And we need to understand what...

Leo: What do you mean, you can't talk about it?

Steve: Well, I'm - there's some things - no, no, no, not my stuff. It's other people's work, some products that are in the works that will be using elliptic curve cryptography, and that's a good thing. So I thought, let's - we've referred to it. We've never plowed in. And, oh, boy, fire up your coffee pot, Leo.

Leo: Is this going to involve math?

Steve: Oh, this is going to - yeah. What I'm going to - okay. That math is so hairy that I'm going to describe it visually and kind of clearly. The point is not to turn us all into cryptographers who could go home and start writing code. The goal is to get an understanding of it, sort of coffee table-style understanding, in the same way where I can say we take two big primes and we multiply them; now it's hard to take them apart. There's a one-way function there. ECC has something - and that's known as a trapdoor function in cryptography. Elliptic curve cryptography has that, too. But it is not as easy to describe. But the flipside of that is we get much more strength. We get something, for example, with 256 bits of, that is, a public key of just 256 bits, we get the equivalent of about 3,000 bits of prime factor crypto, that is, the standard RSA crypto. So much smaller keys, much faster operation. That means smaller packets and easier use. There's implementations for 8-bit chips, 8-bit micros. So there are things that it can do that sort of the traditional public key crypto can't. But it does it because it's a lot more involved. So that's the...

Leo: [Laughing] And that's where we come in.

Steve: And I think I can explain it.

Leo: Before we go to the heavy math - actually, it's not going to be that. If you listen to this show, you're smart. We know that. You wouldn't listen to the show if you weren't. You'll be able to figure this all out. But meanwhile, what's going on in computer technology news, or security news?

Steve: Not too much. We had a big patch from Oracle, our friendly providers of Java, the language many people love to hate, hate to love, have no choice but to love. And many people write to me saying, Steve, I know it's a bad thing, I understand how dangerous it is, but we've got to have it. So for those of you who've got to have it, Oracle just fixed 30 security holes in their various Java packages. They are now at - if you have moved to major v7, they are now at Update 9. If you are still at major v6, they are now at Update 37.

Now, Apple is independently updating Java. But after the catastrophe they had earlier this year with hundreds of thousands of Macs being compromised as a consequence of a known hole that they were slow to patch, they will likely - I think we will probably be having an update from Apple's Java shortly. Nothing so far. This was just yesterday, so what's that, October 16th, that they released this major 30-security-hole update. So you can go to Java.com, check your version. If you need it, the advice is never probably going to change, which is, if you know you don't need it, remove it, if you have it. If you're not sure if you need it, then you probably don't, so remove it. And only if you know you need it should you keep it, in which case you definitely want to keep it up to date.

It is the most routinely exploited entry point for malware. It's just - it's getting less attention because it's not the browser itself. It's right up there with Flash in terms of being an exploit target for hackers. So both of those things being add-ons, large add-on libraries to the underlying browser technology. So it's important to keep it updated. And I

have not seen - I'm a little bit behind, and I'm waiting because I'd like to sort of see when this thing's going to update. And I have it disconnected from my browser, so I'm not in danger. And I've got NoScript locking down my browsing. And I generally don't go anywhere on the Internet that's sketchy. So I'm, like you, Leo, am generally keeping myself pretty safe. But it is a way for bad guys to get in, so time to fix it.

Leo: Yeah.

Steve: We talked, was it last week, I think it was just last week, maybe the week before, about the UPEK fingerprint software disaster.

Leo: Last week, yeah, yeah, yeah.

Steve: Yes. And the good news is they were relatively quick in addressing the problem. They're doing a little bit of finger pointing. But I found a blog posting...

Leo: Wait a minute. Wait a minute. Our software is totally broken, but it's not our fault? Well, who can they point a finger at?

Steve: Yeah, well, okay. So here's what they said. And this is Miroslav Buran. He wrote, "AuthenTec takes security seriously, which is why we contacted ElcomSoft shortly after their recent blog post which claimed that Protector Suite stores Windows passwords insecurely." Now, let's remember that, with just that information, that claim, what we shared last week was an independent researcher who also cracked it. So it wasn't apparently that difficult. Anyway, going on with the blog posting, "AuthenTec evaluated ElcomSoft's claims after they provided relevant information to the company. Based on the findings of our team, ElcomSoft confirmed passwords stored in Protector Suite were not 'barely obfuscated' as written in their report." Okay. So AuthenTec is saying it wasn't as bad as they said. Okay.

"ElcomSoft has reverse-engineered the AES key-generation algorithm in Protector Suite and written code that uses this information to unlock the AES-encrypted storage." Now, okay, wait again. We know that this wasn't - they're saying "AES" like it's some badge of honor. We know that it was dramatically watered down AES-56, which doesn't really exist in the world. They just zero-padded it in order to weaken it, apparently so that it would meet 56-bit maximum key length export requirements. So it's like, okay, well, so, yes, all this is true, but it's not like what AuthenTec had was AES in any standard strength. And then their final butt-covering is, "Any tool that uses this code maliciously must be downloaded by a user and given administrator rights to be effective..."

Leo: So there.

Steve: Well, of course that's true for everything, "making it no more or less potent than widely available keyloggers in harvesting personal information." And that's not true because the login process is specifically protected from eavesdropping with all of that extra stuff that Microsoft does that, for example, keeps - there's no keylogger running at the time that you're logging in, and it is very well protected. And what a reverse-

engineering of AuthenTec's insecure technology yields is information at rest access. And being able to access information at rest is vastly more effective than having to try to get something going by. So it's really not comparable.

Anyway, they said, finally, "In order to protect Protector Suite users" - okay, now I guess they're saying they need protection, even though what they've said before was, well, it's not really that big a problem - "in the event that ElcomSoft makes this code more widely available," which is not the concern anyone really has because ElcomSoft isn't likely to do that, they don't need to, other people have independently broken it - they said, "AuthenTec has created an update to Protector Suite with a hardened version of our AES key-generation algorithm." We don't know what that means. "You can download the update from the link below: support.authentec.com/PS" - as in Protector Suite - "update2012." So that's all run together, one line, support.authentec.com/PSupdate2012.

Now, I have been unable to get to the support.authentec.com site all morning. Fortunately I already had this text grabbed and saved, or I couldn't have grabbed it for the podcast. So I don't know what their problem is. But maybe they're under attack. Who knows.

Leo: [Indiscernible].

Steve: Yeah. Then they said, "Latest version of Protector Suite 2012 including the hardened AES encryption can be found in the download section of the support site: support.authentec.com/Downloads/Windows/ProtectorSuite.aspx. Users of Protector Suite with the Store To Device option available and enabled would not be affected, as keys are stored on the fingerprint sensor and are unique to each PC." So what that says is we wish that everybody was storing their keys in the sensor. But apparently store-to-device option is not always available or enabled, and in which case we store them insecurely in the registry.

So this is an unsatisfying response from a security company. As we know, the proper response is tell us everything about what you're doing. If you cannot tell us everything about what you're doing, then we have no reason to believe that it is secure or that even you think it is secure. So they're still hiding what they're doing. Still, whatever they've done, presumably, is better than what they were doing before. So if you are a laptop user, if you are using UPEK's fingerprint software, and UPEK has been purchased, as we know, by AuthenTec, then I would certainly suggest having the 2012 update running. So there is something now for people to do. Still it's not very satisfying for a company that is attempting to provide very mission-critical security technology.

Leo: I just - it sounded like a non-denial denial. And I just, like, come on.

Steve: Yeah, yeah.

Leo: Take responsibility.

Steve: This is not the right way to handle this at all. Blaming the people...

Leo: Who discovered it.

Steve: ...who found the vulnerability and then accusing them of maybe releasing it in the future, so that's why you're going to increase your security.

Leo: Loathsome.

Steve: Instead of just fixing it.

Leo: Yeah.

Steve: Now, I wasn't sure we were going to have a podcast today, Leo.

Leo: You said that. And I'm puzzled because you are the man. You've never missed one.

Steve: Never missed one.

Leo: What could possibly, possibly make you want to miss a show?

Steve: Came very close.

Leo: [Laughing] I've got to know. What is this magic thing?

Steve: One of my favorite authors, one of this podcast's favorite authors, Michael McCollum at Sci-Fi...

Leo: Ah, Sci-Fi - go ahead.

Steve: ...scifi-az.com, his web page right now, if you go there - maybe you should go there before anybody else does, Leo, if you want to show this...

Leo: Hold on, everybody stop right now. We have a habit of bringing stuff down. Okay, got it.

Steve: Scifi-az.com, you'll see a picture there.

Leo: "Euclid's Wall," Michael McCollum. What is that?

Steve: "Euclid's Wall." Well, Michael sent me his manuscript when he was the only person who had read it. I have since been in communication with him...

Leo: Brought it down.

Steve: ...and he and his wife are reading it.

Leo: Eh, just killed it.

Steve: Okay.

Leo: You're just - I'm glad I pulled it up before you did. Dead, dead, dead.

Steve: I couldn't put it down.

Leo: Really.

Steve: It is really fun. There are so many fun things about it. First of all, I was a little worried because it had a three-masted sailing, I don't know what you'd call it, a galleon or, I mean, like the Nina, the Pinta, the Santa Maria-style big...

Leo: Ahoy matey, yeah.

Steve: ...ship on the front cover.

Leo: Right on the front, yeah, yeah.

Steve: And I thought, okay, you know, I don't know, what is this going to be? And in the prologue, two PhDs basically destroy the world.

Leo: That's not good.

Steve: Now, I would never give away anything important, so that is not an important thing to know because you learn it in the first two pages.

Leo: Oh, okay.

Steve: That sets us up for a really interesting sort of adventure story set further in the future. This is 2087. Two PhDs think they have discovered an infinite source of energy.

Leo: Oh, no. Not the old black hole infinite source of energy problem.

Steve: Exactly. And it doesn't turn out quite the way they...

Leo: Oh, dear. Sucks the whole darn Earth right in there.

Steve: Well, no. But I won't talk about what it does.

Leo: It's not good, obviously.

Steve: But suffice to say it shakes us back to almost no civilization. We then pick up the story a hundred years hence, where there are things like techno archeologists.

Leo: Right, digging this - trying to figure out how did they make these things.

Steve: Who are digging - uh-huh. And little really intriguing bits of stuff that have survived. And anyway, I don't want to say any more. I don't know how soon Michael is going to have it available. He publishes - I don't know if he'll publish in print as he has his other novels, or eBooks, which he has both on his site and on Amazon through Amazon's Kindle.

Leo: I like this genre. Did you ever read a book called "Earth Abides"?

Steve: No.

Leo: Guy wakes, well, a guy goes up on a hike into a mountain. And when he comes back, everybody's dead. Some disease hit. He got a little bit bit, but he survives. Kind of like "The Stand," where a disease wipes out civilization. I love that kind of story.

Steve: Well, this has got romance and good character development.

Leo: Sounds great.

Steve: And the really - Michael always sets up really intriguing problems, which is why I've, I mean, he wrote the Gibraltar Earth series. Just everything he's done I have enjoyed. And anyway, so the point is I just couldn't stop reading it.

Leo: You couldn't put it down.

Steve: I could not stop reading it. And I finished it last night, fortunately, and so we have a podcast today. Otherwise, I don't know.

Leo: Is it a long book?

Steve: No. I got it from him middle of last week, and I had a lot of things going on with me, and I apologize for not having been able to get to it. And so mostly I read it over the weekend and yesterday and really enjoyed it. So I can't recommend it yet because...

Leo: Can't buy it yet, yeah.

Steve: Can't buy it yet. But he said coming next month. I don't know - that's what his web page says right now. So it will be soon, so we're not teasing people too much. But anyway, it was very fun.

Leo: Can't wait.

Steve: Oh, and his other ones, for example, have been trilogies. This one wraps it. So this is not - I would have a different position if I was left with a cliffhanger because, as we know, it's so annoying to, like, read something, and then it's like, oh, god, now I've got to wait for...

Leo: Hate it when that happens, yeah. Good. I can't wait. That's exciting. I'm going to have to go get a copy.

Steve: Now, many people have asked for a third part of our Over the Sugar Hill series, Leo, which I think we should do, but I wanted to formally make a call for our listeners' experiences. It's been more than six months for me. It's been more than six months for everyone. So there is a feedback page on the health-related research pages at GRC, GRC.com/health/lowcarb.htm, or actually /health/feedback.htm. I would like to have - I don't want to just relate my own history and yours. I would love to have our listeners give us a larger sampling of their experiences either way - good, bad, tried it/fell off of it, tried it/loved it, and so forth. So I wanted to formally make a call for people to send me their experiences, let me know if I could use their name or not. I'm happy to keep it anonymous if they'd rather that. It's a little more fun if we have a name attached, but I certainly understand if somebody...

Leo: They're afraid Dr. Mom will find out and...

Steve: [Laughing] And so...

Leo: Don't want that to happen.

Steve: And so we will come back to the topic and have a much broader base of experience to share.

Leo: Good. Can't wait.

Steve: I got an interesting question from - I'm not sure he's a listener, actually, because it just sort of - it didn't specifically say. But he asked a question about SpinRite that I thought our listeners would find interesting. His name is Allan Levene. And he said, "We work in the VDI" - the Virtual Desktop Infrastructure - "and SAN space. Can you let me know how many read/writes your Level 4 SpinRite product does in an hour on a 500GB disk and, if different, on an SSD? I know that you" - oh, he is a listener because otherwise he wouldn't know that I only recommend Levels 1 or 2 on SSDs, which of course we've been discussing recently, so Allan is probably listening to this. "But some expensive ones are claimed to have a very long life in spite of whatever you throw at them. I'd like to find out if that's true."

Okay. So I wrote back, I said SpinRite's surface analysis/pattern testing strategy has evolved significantly since the early days of MFM and RLL drives. When drives switched to PRML encoding - which is what they use now, that stands for Partial Response Maximal Likelihood, which gives you a sense for how far out into voodoo weeds...

Leo: I don't want to hear "likelihood" when it comes to my hard drive.

Steve: Exactly.

Leo: There might be data there.

Steve: Partial Response Maximal Likelihood. The bits are truly so close together that they are now interacting with each other. So the drive looks at what it gets back when it's reading and chooses the highest likelihood of what was written that would result in that being read back. I mean, that's how far out we have gone. I mean, it really has become amazing that these things work at all. Anyway, and so what's happened is there are several stages of processing between what I would call the "user data," that is, the data we think we're storing on the drives, and what actually is written. There was, for example, back in the early days, you will remember, Leo, there was something called "write precompensation."

Leo: Yeah.

Steve: And what happened was, at a certain cylinder of the drive, write precomp would be engaged. And what that meant was, think about it, as you go to successively inner cylinders that have a shorter circumference, but you're storing the same number of bits on each cylinder, that is, on each track, if the track is further in, it's got an actual lower circumference. It's a shorter track in terms of its linear measurement. Which means the bits are pushed closer together.

Well, it turns out that, when you read back bits of differing polarity which are close together, they read back as closer than they actually are. That is, they appear closer than they were written because they are influencing each other. So write precompensation, it precompensated for essentially sort of that bit attraction by writing them further apart on purpose so that, when they read back, they would be in the position that they were to be expected.

So this gives you - that's the old days. But so that's an example of some algorithmic fudging that was being done in order to allow more bits to be stored. Well, we're now decades past that, where maximal likelihood is the domain we're in. The point is that SpinRite once knew how to write patterns that were so-called "worst case patterns." Those were patterns which deliberately used the knowledge of what was difficult to read in order to make it harder for the drive to do its job, thus those were patterns that SpinRite used for testing the surface and testing the whole reading/writing channel.

Well, there's no way to know today what we tell the drive to write and what it actually does write because there's several stages, about three or four stages of preprocessing that go on. There's a process called "whitening" where, no matter what we write, we end up with a 50-50 statistical probability of bits, and all kinds of prewriting encoding in order to put down flux reversal patterns that again have the maximal likelihood of being read back. So this notion of pattern testing has really gone away. Consequently, what SpinRite does is something that is simple and fast because the other thing that's happened, of course, is that drives have gotten huge, so you just can't afford the time of doing 30 different test patterns on a drive.

So SpinRite reads the data, inverts all the ones to zeroes and zeroes to ones, writes that back, then reads that and verifies that it got back the inverted data. Then it reinverts it, or, that is to say, rewrites the original data and then rereads it to make sure that it could read that back. So it says, okay, there's no way to know what's actually being put on the drive. We've going to verify that, at the data level, we can write ones and zeroes to every location in terms of the users' data, and we will use the tremendous capability of the drive to recover whatever it's got back to our data and use that. If there's a problem, the drive will see it and then will relocate that data to safety, as we've described in the past.

So consequently, SpinRite writes exactly twice the size of the drive while it's running, if you're running in the Level 4 pattern-testing mode, where it's actually writing to the surface, even if it was able to read it without any trouble. So SpinRite reads the surface, recovers the data, then does its dual inversion, flipping all the bits from zeroes to ones and ones to zeroes twice, and verifying that the drive was able to read that back in each case, which again shows the drive if it's going to have a problem with that area in the future.

So if you had a 2GB - 2GB! - a 2TB drive, SpinRite would write 4TB of data. In the case of Allan's question, a 500GB disk, it would write 1TB worth of data, essentially writing

every sector on the drive twice. So that's the answer to the question and another little bit of peeking under the covers at how SpinRite works.

Leo: This is good. You put this all together, you have a pretty good idea. Very impressive. Lot of energy, obviously.

Steve: Lots of technology under the cover.

Leo: All right. Beanies on.

Steve: Okay. So we've talked many times about symmetric cryptography, where you have key lengths of 128 bits, 192, 256. AES, our current symmetric encryption standard, is that. It's very fast. It's used for so-called "bulk" data encryption. And things like RC4 is another example of, in that case, a stream cipher which is used for performing bulk encryption. So we have those which use private keys or secret keys which are able to encrypt and decrypt at high speed.

We also famously have public key cryptography, sort of the flipside, where instead of using one key, which is used typically with either one or two algorithms, you may have, for example, in the case of RC4, it produces a bitstream which we exclusive OR with the data in order to encrypt it, which means we do the same thing to decrypt it. We produce the same bitstream over at the decryption end and exclusive OR it again, which turns all the bits back into what they originally were.

There's an example of one algorithm that is used to do both encryption and decryption, or typical iterative crypto like AES. We did a podcast on exactly how AES works, where you take the key, you run it through a so-called key expansion in order to create lots more bits than the key has, and then you successively run your decrypted data through an iterative process that uses chunks of the derived key bits each time, and you run it through, like, 11 or 14 or however many times the cryptographers determine is necessary to sufficiently scramble the data. And every time through that iteration it takes a unique input combination and maps it, essentially, to a very difficult-to-track output combination, yet in a single iteration doesn't do it enough that you couldn't figure it out. Thus they run it through enough times that the relationship between bits is lost from what you initially put in and what you initially get out. But that means that decrypting it, you can't do the same thing. You have to literally reverse the process.

So with a cipher like AES, that is to say Rijndael, you decrypt with a different approach. You run it sort of backwards in order to get back what you put in. So that's the whole symmetric side - relatively fast, a secret key. You use the same key, but either the same or different algorithms, depending upon the way the cipher works. With public key cryptography, we have a whole different model, and elegant in its own right, where we have typically a private key and a public key. And it doesn't have to be, the other key doesn't have to be public, but the idea being that one key is used for encrypting or enciphering to produce something unpredictable given an input and you get an output. And only the matching key, the key that was made - typically these are made as a pair. One is kept private; the other is made public. But again, only the other key can be used to undo what the first key does.

So that was a huge breakthrough for crypto because, until that time, there was the whole problem of, well, if I have a secret key cipher, like we've talked about first, like AES, how

do you get that to the recipient securely to allow them to decrypt what you've encrypted? So there's a whole communication problem. You need some sort of a secure channel to get the secret information. With public key cryptography, that problem is solved because you can just pick a big random number, and you can encrypt it with your recipient's public key and then send it publicly because only somebody who has the matching private key, and presumably the recipient would be keeping it secret, is able to decrypt what you encrypted with their public key.

So this is the foundation of all of our modern crypto that we've talked about. And the idea is the public key technology cryptography, as I mentioned at the top of the podcast, it works, but it works in a fundamentally different way, where we rely on some sort of a difficult problem, but it's sort of a softer domain. That is to say, for example, with traditional public key crypto like RSA style, where we rely on the difficulty in factoring the products of large primes, well, it's the largeness of the primes that creates the problem. If these were small primes, then their product would be small, and computers are so fast these days, that it wouldn't be that difficult. We have spent a lot of energy on coming up with ways to produce prime factorizations of numbers. So it's not that that is difficult so much as it is time consuming. And we've never, despite all the time and attention we've given to it, we've never found a way to short-circuit essentially what is brute force.

So the strength we get from traditional prime factorization style public key cryptography arises from the size of the factors, the size of the numbers. But the flipside of that is the keys must be big. So everything must be big numbers. And so what that means is, when we're dealing with big numbers, then processors still have word sizes much smaller than those big numbers. And we're talking about 2048 bits. But the processors are down at 32 or 64 bits typically. So that means they have to do everything in multiple chunks, and they have to handle multiple precision math and really hairy stuff.

So what that means is that that cryptography is not applicable for bulk crypto. That's why we use the clever approach of choosing, as I mentioned before, we choose a random number. We encrypt that using the public key technology, send that to someone who can decrypt it. Now we've essentially managed to share a symmetric key which we can then use for performing our bulk crypto. And that's basically the way SSL and many of these other hybrid protocols function.

Now, everything's fine, except that it's difficult to put this kind of very processor-intensive technology on a smart card or on an RFID tag...

Leo: Ah.

Steve: ...or on a near-field chip. Did I hear you go "Ah," Leo?

Leo: Yeah. Now I'm getting it. Why do we need another one? Well, because this one is computationally intensive.

Steve: Yes, exactly.

Leo: So we need a way we can do it less expensively.

Steve: Exactly. And what would be really cool is if we had a technology that was public key crypto that was lightweight enough that we could use it with near field technology. That is, something like printed on a piece of paper. The weakness, as we discussed when we were talking about near field stuff, is that it's just basically giving you a serial number, like an RFID tag. And now there are better solutions for that. But we're never really talking in terms of serious crypto. For future applications it would be nice if we had stronger, that is to say, really good public key-style crypto for something that could be powered, I mean, that is computationally simple enough that it can be powered off of a pulse of electromagnetic radiation rather than needing heat sinks and fans to keep it cool while it's doing its work.

Well, enter elliptic curve cryptography, ECC. It's an entirely different way of solving this, essentially giving us the same feature. And because it's got nothing to do with prime factorization, it's like starting over. It's like, okay, let's come up with a whole different approach. So as I also mentioned at the top, it's not my intention to turn us all into people who can go home and write code for ECC at the end of the this. But I want to give everyone sort of a level of comfort, a feeling for kind of what it means so that it's like, okay, I kind of get that. Sort of at the same level as, okay, I understand if I take two big prime numbers and multiply them together, then it's going to be difficult to take them apart. Multiplying is easy; factoring is hard.

So what we have with elliptic curve cryptography is a dip into some kind of hairy math, but not impenetrable. We just sort of have to say, okay, I'm going to allow this. So, for example, so what is a curve? A parametric curve is what elliptic curve cryptography uses. Now, a simple parametric curve is x = y. And so if we took a two-dimensional - and this is planar. This is all just two-dimensional. So we have just an x and a y axis that are orthogonal on a two-dimensional plane surface. So the parametric curve x = y is, as we know from our high school algebra, is just a straight line, a diagonal line running at a 45-degree angle because, if x = y on our graph, where we have x = 1, y will be one; or x = 2, y will be two; or x = 3, y will be three. And all values in between, similarly, x and y are the same. So this x = y equation describes this straight line on our graph.

Now, if we added a constant, say y = x + 5, well, so now, if x = 1, y = 6 because we have x + 5 = y. So if x = 1, y = 6; x = 2, y = 7. So again we have a diagonal line, but it's been shifted upwards by five. So it's the same 45-degree angle shifted up by five. Now, if we were to multiply one of these parameters, say y = 2x, well, then when x = 1, y = 2; when x = 2, y = 4; when x = 3, y = 6. So what's happened is the slope of the line changed when we added that two parameter to the y = x, so it's y = 2x. Now the slope has increased. If instead we said y = x/2, which is the same as x(1/2), then the slope of the line decreases. It's more horizontal. It's flatter. And similarly we could do both at once. We could say y = 2x + 5, which would give us a steeper line, which is also moved up by five. So we can do both at once. So this is something we all obviously know from high school algebra.

Now, we can also do something a little different. We could say, for example, y = x^2, or x(x). So now we get a different - now suddenly we don't have a straight line anymore. If x is one, then y equals 1x1, which is one. If x is zero - we ought to start there, maybe - then y is zero. When x goes to two, though, y jumps up to four because we have y = 2x2 is four. When x is three, y jumps up to nine because we have 3^2. So what we get is a parabola. We get a curve on the positive side of x which goes up very steeply. And on the negative side of x it also goes up very steeply because -2, if x is -2, minus two times minus two is positive four. So thus we get a parabolic curve. We could then do things to it like add or take factors and so forth.

Okay. So I wanted to give us a little bit of foundation first, just so we're on the same

page, because the equation for elliptic curve cryptography, all ECC, is y^2 = x^3 + ax + b. That's all there is to it. So y^2 equals x^3 plus ax (where a is a parameter) plus b (where b, as we saw before, skews the whole thing). That is sort of the master equation for elliptic curve cryptography. So what that describes, this y^2 = x^3 + ax + b, it's a complex relationship between x and y. So depending upon what the parameters are, it can look very different. The concept, then, is that we have points on this curve which we multiply by themselves.

So think about we have this, if you imagine a two-dimensional surface with some lines drawn on it, and the lines satisfy that equation. That is to say, so there are continuous curves, but every point on that curve, if you take the x coordinate and the y coordinate, that equation is satisfied for chosen a and b parameters. So the act of coming up with turning this from that curve to crypto is we have, again, I use the term "trapdoor." That is, a trapdoor is a one-way function. It's something which we can do easily in the forward direction, but we cannot do it easily in the reverse direction. In the case of prime numbers, the trapdoor is it's easy to multiply two primes; it is difficult if you don't know what they are, if you only have the product, to pull them apart again. So that's the trapdoor function upon which traditional RSA resides.

In the case of elliptic curve cryptography, we have a curve with known parameters, that is, when these algorithms are published, or when the implementation of this is published, the parameters a and b are public. They're known. And they are the subject of standards. NIST has a bunch of ECC curves where they've worked out the details, and the a and b parameters are part of what is described as, given this curve, then this is the one that we will use to perform our encryption. So the process of encryption is to take a point on the curve and multiply it by itself, that is, through successive addition, where we take a point on the curve and add it to itself.

Well, now, it's like, okay, wait a minute. It's a point on the curve. How do we add a point to a point? Well, this is defined by the mathematicians as something called "point addition," which exists in the field of math for these kinds of curves. Point addition is defined as taking two points along a curve and computing where a line which passes through them intersects the curve. So you have two different points on the curve. And again, this y^2 = x^3 + ax + b curve, two points on there. So those two points, as we know, again from high school algebra, two points define a line. So somewhere else the curve intersects this line. So the point of the line's intersection that passes through our two points that we're adding, where that line intersects the curve, that's what's called the sum of the first two points.

Well, okay. The problem is how do you start because, if you're starting with just one point, then you have a whole family, essentially, infinite number of lines because there is no second point different from the first to establish a line. So the mathematician said, oh, that's easy. We will define the straight line where the two points are the same as the tangent line at that point in the curve. So we start with a point, and the curve will have some slope at that point. And so we obtained a tangent line, which is of course the line that runs tangent or perpendicular to, well, has the same slope as the line, our elliptic curve line, does at that point. And so that defines the line for the first two points which are not different from each other. That gives us a third point. Now we've got two points, and we can then draw lines between them. So if you're still with me...

Leo: I got it. It's a tangent.

Steve: Yeah. So we take a point, and we find the tangent angle at that point. Then we

look at where that line intersects the curve, and that gives us our second point. Now we have that point and our first point. And they form a line which we then find out where that intersects the curve. That gives us our third and so forth. So this allows us to iteratively sum from a given point some number of times. In elliptic curve crypto the number of times we do that is the private key. So we take a random number. We take a random number that is going to be our private key, and we multiply, essentially - think about it. If we take the point plus the point plus the point plus the point, well, that's n(p) where n is our private key. So we're multiplying the point by our private key.

The result of all of that, and you can imagine, like, how constructive, sort of geometrically constructive this is because we're dealing with this wacky elliptic curve and plotting points on it and where lines through the pair of points intersect the curve, and then we jump to there and continue, we do that however many times is required for the private key that we choose, the result is the public key. And the point is there is no way, you can well imagine, no way to reverse that process. That is, this is a very difficult one-way process. The trapdoor function is doing all of that from a given private key is the number of times we move along this curve, computing these points, and the result is the public key. And it is possible using this math to essentially, well, what we get is the same sort of thing, a one-way function. We can easily or relatively easily go forward. There is no known way to go backward.

And the reason we've gone through all this is the bit space that all of this operates in is vastly smaller than the bit space required for factoring. Factoring is not difficult enough that we can use small numbers. We have to use huge numbers because we're pretty good at factoring. This wacky system that we've come up with, with elliptic curves, is understandably or believably difficult enough that relatively few bits gives us equivalent protection. 160 bits of ECC is the equivalent of 1024 bits of traditional prime factorization public key technology. Remember that 1024 has been plenty strong for a long time. I only recently, and everybody else moving to extended validation certificates, only recently went to 2048. And that's believed to be strong forever. So, I mean, like a long time. So most of the websites on the 'Net are still at 1024 bits of standard RSA encryption, and they're plenty strong. We get that equivalent strength with just 160 bits within the ECC space.

Leo: So why don't we use it for everything?

Steve: It's just - it's newer.

Leo: Ah.

Steve: Cryptographers are famously conservative. I mean, look at here.

Leo: As they should be, yes.

Steve: Yes. We launched the SHA-3 competition, what is it, eight years ago just because maybe SHA-256 wasn't going to be strong enough. And it turns out, oh, it is strong enough. We really don't need SHA-3. But cryptographers are super conservative. When I was researching this to get a better sense for where we are, there is an annual - there is a bunch of annual conferences that are held, and there's someone who sort of maintains

a state of ECC. And he said, well, for the fifth year in a row, I'm here to report that really nothing has happened.

Leo: So this is how old? Five years old, then?

Steve: Oh, no, no, this is a couple decades old.

Leo: Oh, all right. So it's not that new.

Steve: Exactly. Really it was probably more than anything the strength of RSA, just the political strength of RSA. Everyone was enamored of the idea that they got all the press and the attention. RSA's patents expired in 2000. The original designers of this put it into the public domain, but there have been some implementation tricks that were patented. So this isn't - ECC is not as obviously free of intellectual property collision except that a lot of people are taking the position that it is, in fact, that you can do everything you want to do with ECC today with no intellectual property concern.

Dan Bernstein, whom we've spoken of, is a famous cryptographer. He's got a bunch of ECC stuff on his site and free open source. It is now, for example, it's in OpenSSL. It's in the Crypto++ library. It's in the GNU TLS library. It's in OpenSSH, the NSS. The Mozilla Security Suite has had it in their crypto libraries. Now, remember the way SSL operates during that initial handshake. This is one of the nicest things about the way SSL and TLS was designed is that the client says, here's all of the crypto suite tools that I know of. And then the server is able to look among them and choose the strongest of those that it also supports. So they automatically handshake and negotiate for the strongest means of establishing a relationship.

That means that, for example, when you build an instance of open SSL, and you enable the ECC cipher suites within the protocol, when you connect to a server that has had the same thing done, you'll automatically negotiate a strong ECC connection. And if both ends support it, you can get by with a much shorter bit string which allows you to essentially short-circuit the long setup time that traditional SSL has when it's forced to use standard large prime public key technology. So what's happening is this is getting attention because performance is becoming a problem. People have been pounding on ECC now for a while. I guess it's less obviously or it's less intuitively difficult to reverse this process.

As of 2009 there is still no proof, actual formal proof of the security of elliptic key crypto, which isn't a bad thing. It's just a hard problem. Similarly, there's no proof that we can't factor large numbers into their primes. We just have - no one has ever figured out how. I mean, they've gotten better at it. But still, if they're big, we cannot figure out a way to do it. We've never been able to prove we can't. But it seems unlikely. And we're resting our entire foundation on the fact that we can't. So the cryptographers are actually more comfortable with elliptic curve crypto than they are with prime factorization crypto. That is to say, the people that have been looking at this now, they feel better about the strength of elliptic curve crypto in the future than they do about the future strength of crypto relying on prime factorization. The sense is everybody's a little uncomfortable about how strong that actually is.

And the other thing is that, as people have been working on both, the sense is that the prime factorization problem has been softening at a faster pace than the elliptic curve

crypto problem has been. So the elliptic curve, the people working on it feel it's harder, it's a harder problem. It's less likely to fall to some breakthrough than factorization is. And everybody likes the fact that you can, with 160 bits, you can get the equivalent strength of 1024-bit prime factorization-style public key technology. And so what I think we're going to see is at some point in the future we will be talking about devices which are using elliptic curve crypto, and now we all know what it is.

Leo: You are masterful, my friend. I don't understand a word you said, but it is masterful. It's really interesting. And it sounds like, in time, it might replace traditional public key crypto.

Steve: I kind of think it's going to. Again, cryptography and cryptographers are conservative. And we're all glad they are because, when you think about it, there have been mistakes made in the past with the application of these things. Even RC4, that was famously troubled in its initial implementation with WEP, the Wired Equivalent Privacy encryption used in the first WiFi protocol, well, it wasn't RC4's fault. The implementers of the RC4 algorithm did some things like not letting it warm up enough. It generates pseudorandom bit sequences, but you've got to let it churn for about 256 cycles in order for it to have scrambled itself up enough, and they didn't. They immediately started using the first things that it emitted, and it turns out that what that did is that meant that it hadn't scrambled up the key that it was given. And so there was a findable relationship between the key and the pseudorandom sequence that was being used to XOR the data, and it fell as a consequence.

So what's nice is that we really don't have any instances of an outright collapse. We have things like MD4, which, okay, no one ever broke it, but we got so much faster with our GPUs and our computers that it no longer was something we felt comfortable relying on. It became soft, and so we moved to MD5. And then that got a little soft. And so now we're at SHA-1, but new things should use SHA-256. So what we're seeing is we're getting better at understanding these problems, the problems that we have deliberately erected for ourselves in order to create privacy. And we're staying, however, well ahead of the, if you'll pardon the pun, the curve.

Leo: I suspect, just because it's so computationally simple, and we are doing so many new devices that need crypto, that this really will be a great choice.

Steve: Yes. And the idea of having a lightweight, inexpensive technology which can be deployed with strong public key style technology, not just private key stuff - well, for example, all of the dongles that we've been talking about, those are all symmetric key. Those have a secret in them.

Leo: And that's insecure.

Steve: Yes. Exactly. There is a secret in them, and we're relying on the fact that they're just giving us the output from that for the security. But there's no sort of a challenge, we're not giving them a challenge that they have to then answer to prove that they are who they are. They're just - we're maintaining a secret at RSA and a secret in the dongle, and we've already seen what happens if RSA is unable to keep their secrets. So we're going to take a next step here before long, and I wouldn't be surprised if, at some

point, certainly within the life of this podcast, which seems to show no sign of...

[Talking simultaneously]

Steve: ...we'll be talking about some cool stuff that have ECC in them, elliptic key crypto, and now comfortable with the concept because it's going to be secure enough for us.

Leo: Very interesting. Really great stuff. ECC. Not the ECC you're thinking of, elliptic curve cryptography.

Steve: Yes. Not error correction code. That's a different topic.

Leo: Steve Gibson knows all about that, too. And he will no doubt, in fact I think he already has talked about that on a show. We have lots of them, 374 shows in toto, available in two places, one on Steve's site, GRC.com, and he does 16Kb versions of the show and transcripts of the show. So that's really great, if you want a small, compact way to listen. We make video and high-quality audio available at our site, too, TWiT.tv/sn for Security Now!.

When you get to GRC.com, though, check out SpinRite, the world's best hard drive maintenance and recovery utility. You've got to have, if you've got a hard drive, you've got to have SpinDrive [sic]. And lots of great freebies, too, which he's always giving away. Next week we will have a Q&A, I presume, unless major breaking news happens over the next few days. And if that's the case, you can ask your questions at GRC.com/feedback. Steve will pick 10 good ones for next week's - did I say "SpinDrive"? No, SpinRite. SpinRite.

Steve: And I do want to encourage people, if you've had a story that you think would be interesting for us to share with your Very Low Carb experience, I know people were sharing initially. I'd like to know how that's worked out over the course of six months. And we will do a third, I think on a Saturday after your Tech Guy...

Leo: Yes, that's the best time to do it, yep.

Steve: ...we'll sneak one in and let everyone know that it exists.

Leo: Saturday or Sunday. Actually might be better Sunday.

Steve: Oh, because I thought that collided with TWiT.

Leo: TWiT is an hour after The Tech Guy. So we did it last time, I think we did it on Sunday.

Steve: Oh, yeah. We can easily. Cool.

Leo: Yeah. But we'll let you know ahead of time.

Steve: Yup.

Leo: And no, Gyver [ph], we're not going to do a Very Low Carb podcast. I think we'd run out of things to say. I think there's a paleo show somewhere, though. Is it 5by5 or somebody has a paleo show.

Steve: Oh, there are several, yes, podcasts.

Leo: You could go listen to those. I absolutely encourage it.

Steve: That's too much.

Leo: Hey, thanks, Steve. Go ahead.

Steve: Thanks, Leo.

Leo: But do you have something else you want to say? I don't want to interrupt.

Steve: No. No.

Leo: All right.

Steve: Glad you stopped me. Go have some lunch.

Leo: We'll have some lunch. All right, Steve. Thanks a lot. See you later.