Chatting Off the Record With OTR

Description: After covering the week's security news, Steve and Leo examine an interesting security protocol known as "Off The Record" (OTR) which has been specifically designed to protect conversational privacy, both as it happens and also in the future.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-406.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-406-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. And coming up in just a bit, we're going to talk about a new chat protocol that's absolutely secure and private - OTR - and an implementation from the Cypherpunks called Cryptocat, plus all your security news. Security Now! is next.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 406, recorded May 29th, 2013: Off The Record With OTR.

It's time for Security Now!, the show that covers your security online, your privacy online, science fiction, coffee, yerba mate, and anything else that is going through the brain of this man, our Explainer in Chief, Mr. Steven "Tiberius" Gibson. Hello, Steve.

Steve Gibson: That would be kind of cool, if that were my middle name.

Leo: You can take any middle name you choose, and I think you should take that name in honor of Captain Kirk.

Steve: I think that would be okay. So we have a neat episode. We've in the past talked a lot about the fundamentals of security, sort of the building blocks. And, for example, we often refer to PGP as a popular means of protecting, for example, email communication, Pretty Good Privacy, Phil Zimmermann's famous protocol. But there are some problems with what it does and the way it works. For example - well, we'll get into that. But what I want to talk about this podcast is the protocol that under...

[Talking simultaneously]

Leo: Underlies. Under...

Steve: Underpins, underpins.

Leo: Underpants, yes.

Steve: Yeah, well, we knew it was under.

Leo: Okay.

Steve: Beneath the Cryptocat web chatting service, which is known as Off The Record, OTR. And Off The Record is really what I want to talk about, which is more interesting. Cryptocat is a super simple, low friction, easy to use, easy way to get into this concept of OTR, Off The Record, protocol usage. Yet it turns out that this Off The Record protocol is the key and the cool part of Cryptocat. Those guys, the Cryptocat guys, have just done a nice implementation of it. But, for example, it exists, it is available as a plugin for almost all mature IM chatting clients. You can get it for XChat and for all the rest. We'll sort of enumerate those toward the end. And the way it operates, the way it was designed, and the features it deliberately provides are just, I think, fascinating and form a perfect topic for a security-based podcast.

Leo: No relation to the CueCat barcode scanner.

Steve: None whatsoever.

Leo: No, no. Just Cat's in the name.

Steve: Yeah.

Leo: All right. Good. Well, this will be fun. I have - we just have one proXPN commercial. So why don't we - I know you have security news. So why don't we do that, and then we'll do the commercial before we talk about Cryptocat.

Steve: So probably, I don't think there's been anything that's been more tweeted in a week. Actually it's only been three days. It feels like longer than that. Ars Technica, you may have seen this, Leo, or probably received your own share of tweets about it, did on the 27th a story titled "Anatomy of a Hack: How Crackers Ransack Passwords."

Leo: Oh, yeah. This scared me, actually, because they were talking about how a 16-character basically random password was cracked in an hour. And I just thought, oh, boy, I'm glad Steve Gibson is here to talk about this.

Steve: And thus the reason so many people tweeted, because they wanted to make sure that I knew about it. And absolutely. So in their headline they said, "How Crackers Ransack Passwords Like...," and then they gave the example of "qeadzcwrsfxv1331."

Leo: That's a pretty good-sounding password.

Steve: It's like, yeah, that would - you'd think that would be strong enough.

Leo: Yeah.

Steve: But it fell to these people. So I just tweeted the link to this [ars.to/15iZSgq], although you could also - the URL under the ArsTechnica.com domain is how-crackers-make-minced-meat-out-of-your-passwords, all hyphenated. So I'm sure Google, if you put how crackers make minced meat out of your passwords, would take you to the article, or you just go to...

Leo: Well, also it's right on the front page right now of Ars because it's such a good...

Steve: Well, and that would be it, too.

Leo: ...topic, yeah, everybody wants to read it.

Steve: So a couple things. First of all, it's a three-page read. I recommend it to our listeners because there's a lot of information there. Now, one of the things that we will spot, we listeners of this podcast, first of all, is that they chose to use as an example the weakest possible password-hashing technology.

Leo: Oh, MD5, yeah.

Steve: Yes. So...

Leo: Well, they even say that. They say due to flaws in MD5 this is easier.

Steve: Well, so, yes. I mean, so, yeah, they do explain it.

Leo: Acknowledge it, yeah.

Steve: But that's still what they use. So while acknowledging it, it's really important to understand that these were unsalted. It was the worst hash that nobody uses anymore.

Leo: Although I should point out, I think it was - who was it that was using this unsalted fairly recently? I mean, people do, websites do use it.

Steve: Right. But one of the things that has frightened people are the benchmarks that Ars Technica ran. And so it's important to recognize that all of this is a function of the technology, that is, the technology here is MD5 with no salt and no strengthening, which is to say no iterations.

Leo: And physical access to the password database, the hash database, so that you can run many, many, many brute-force attacks on it in a second.

Steve: Now, now, okay. With that said, though, I mean, there's still a lot of value in the article. So I don't mean to in any way discount the article. It is absolutely worth reading for a couple takeaways that I thought were interesting. First of all, as in these three pages, the author, who interviewed three different hackers - one whose name they never knew, he remains anonymous, using, I don't remember what his handle was, Pixel or something - interviewed them while they were working. They were all working with their preferred cracking tool, which is the open source, freely available, oclHashCat+, which has its own web page and features. It's a multi-parallel GPU. It can scale to 128 simultaneous GPUs, Graphics Processing Units, in a monster system. It doesn't - you don't just, like, load it up and go. You interact with it, which was one of the interesting aspects.

Leo: Although that third hacker was using, like, one kind of a PC with a 7970 card. Radix wasn't using exactly the high-end stuff to crack it; right?

Steve: Well, exactly. In fact, none of them had super large hardware. They did, in fact, they commented at one point that, because they didn't have access to their normal cracking box, their strategies were a little different. They might have been a little...

Leo: Basically normal computers with Radeon cards in them.

Steve: Yes. Yes. None of them were high power. So again, that helps to offset the fact that it was MD5, no salt, and no iteration. So it's pretty much just straight, put a string in, almost instantaneously, well, yeah, I mean, virtually instantaneously, get out the MD5 hash, and then see if it's in the list that you're trying to crack.

So the really interesting thing that the article explains is the strategy that these three guys deploy. They have word lists, so naturally the first thing they do is just run their word list through the hash that they know the site is using, the hash algorithm, which in this case is MD5, no salt, no iterations. That instantly turns their word list into a set of 128-bit, because that's the size of MD5's output, 128-bit hashes, which they then cross-reference with the hashed stolen or leaked whatever password database to see if they get any hits. And, you know, bang. Each of them, because they're working from their own password lists, they get varying numbers of collisions.

Leo: One list has a billion passwords on it. I mean, it's huge. So you're going to get some collisions.

Steve: Yes.

Leo: Because a lot of people use sexymonkey123. That's for sure on that list. And if they used it in this particular database, it'll work; right?

Steve: Well, and, so, yes. So it's worth reading the article because they do a number of things. For example, they'll take their word lists, and they'll just use them like that. Then they'll do the sort of things we would, if we were scratching our head thinking about it, we would try. Like then they add - they try all different four-digit combinations appended to the end. And bang, another couple thousand are found that way. Because somebody says, you know, sexymama4267. And they think, oh, no one's going to ever get that. Well, yeah. That came right after 4266. So that wasn't that hard to find.

But the thing that I really thought was most interesting - oh, and they did make the comment that, in this case, they did not know from which site this list came, probably to protect the site. I mean, we don't know really what the history of this password list, of the password database was that these guys were cracking. But they made the point, obviously, that if you knew where this came from - and normally you would because you're trying to crack passwords for that site. You know that you've got the database from some site. So that's the point is that you would look very closely at the password policy, that is, the password acceptance policy of that site. So if that site says your passwords must be 10 characters or greater, well, they're not going to try any six-character passwords, or seven or eight or nine. That's just - they're not going to.

But they didn't have that in this case. So they did try some - they're, like, I think some of them had, like, every possible six-character password, just bang. And they ran that through the thing, and that took an hour.

Leo: Geez Louise.

Steve: And it found some. And then they concatenated them. They tried word mixtures, and that found a bunch more. But the coolest thing of all was that, once they had found, oh, 6 or 7,000 passwords, they actually run a Markov chain, a Markov modeling against the passwords they have found because what the professional crackers know is that, for a given site, for whatever weird reasons of, like, the psychology of the people collectively going to a given site, the passwords they choose tend to be similar. Which I thought was fascinating. So it's like the site influences the passwords you create to some degree. Or your choice to go there may be, if you're bringing a pre-prepared password that you use, unfortunately, across many sites, who knows what it is?

But what they have found is that, if they analyze the passwords they have easily cracked for statistical differences, the number of characters, the number of uppercase, the incidences, the location of digits in the password, if they look at the passwords they have cracked, it gives them a strong advantage against the ones they have not yet cracked. Which I thought was, oh, that's just so cool, I mean, that that actually works. And so they did that in this process. And so they used, like, their standard approaches to get

thousands of more readily cracked ones. Then, looking at what those were, they then modified the guessing. That is, their results from the early cracking, the easy cracking, fed backwards almost evolutionarily to drive the next generation of guessing.

And HashCat+ does all this. I mean, it's interactive. At one point one of the guys was doing something, and he said, well, that's going to take 20 hours, but this was like a real-time interview mode. And then he said - so he aborted that after a few minutes because it wasn't finding much, or the rate at which it was finding them didn't impress him. And so he said, okay, stop that, and let's go try a different direction. So anyway, terrific article. And thank you, everybody, for sending it to me.

Leo: What's the takeaway on this? I mean, websites obviously need to use stronger hashes than the well-known, broken MD5.

Steve: Yes, yes.

Leo: They need to salt them. But if a web - but you don't control that. I mean, when I log into a site, A, I don't know it.

Steve: Right.

Leo: I might be able to infer it from their policy. But I don't know it. And, B, we don't control what they choose. I can't say to them, well, I'm not going to use MD5. Would you use something else?

Steve: So nothing we saw in this article said that I was wrong about Haystacks. Which is not to say that somebody should use a simple repeating pattern as their entire password. But the whole concept of Password Haystacks was to push you out, to make it easy to make a much longer password than you would normally ever have. Yes, it's true, completely random passwords are best. That's the takeaway. And in any scenario you want as long a password as the site will let you use, and it wants to be completely random. Which means you cannot remember it.

Leo: But it is a hockey stick when it comes to password length.

Steve: Yes.

Leo: And they do show that graph.

Steve: Yes. I was going to have you bring that up ahead of time because that's sort of like, okay, I rest my case on length. They call it "hitting the wall" of password length. As it gets longer, it just goes exponentially straight up in the amount of time it takes to crack these. So that was my recognition, which is why I created the Password Haystacks concept, was start with a good password and then pad it. Make it longer. Because remember, they don't know. They don't know anything about what you've put in until

they find it. And if you make it longer, they're just - they're not going to find it. But the way the Ars Technica article ended was saying nothing beats really long, totally random. Nobody can remember that. So, and Ars Technica said we will shortly be doing a series on password reminder programs a la LastPass and 1Password.

Leo: Does this change how I think about LastPass? Because that's the one password, you know, I have to remember one password. I use LastPass to generate 16 or longer character passwords, random ones, for every site I join. But I do have to remember one password. That's the LastPass password. And for that I make it long, but I use a mnemonic. Is that - should I worry about that?

Steve: Well, and remember, one thing we do know about the LastPass guys is they're on top of this. They're not using MD5 with no salt and no iterations.

Leo: Good point, yeah, okay.

Steve: So they recognize that they're holding a massive asset in the form of all of their customers' passwords. But we're encrypting them before LastPass ever gets them. So LastPass never gets them in the clear. Remember that all the sites we go to, they get them in the clear, and then they hash them to store them. So there's a vulnerability right there. LastPass, it's client-side encryption, really good client-side encryption. And they recently beefed up their PBKDF2 iterations. I think they recommend, is it 500, I don't remember now, but that's something...

Leo: 5,000, I think they said.

Steve: So you're able to set that and crank it up as high as you want. So that happens in your browser. Then it gets sent there. Then they salt it and further mix it and, I mean, because they don't want the responsibility of having all this. So my feeling is, while it is true that we end up with a rather complex and sophisticated chain of security, and we know that that's always dangerous, we're storing really good passwords in a secure manner with something like LastPass, which you and I have both chosen after really looking at it carefully. So Ars Technica's going to follow on this story by essentially going where we've already gone and saying the best passwords are long and random. So if you can use those, you should, and you need to use some sort of password minder to manage that for you.

Leo: LastPass does so many great things now. It's gotten better and better. They've even got - if you put a credit card number in there, they'll do, for free, you get some credit card watching features. They remind you now - and this has become an annoyance, but it's a good thing. If it says, hey, you know, you use the same password on this site as you use on other sites, they'll let you know now. Which is great.

Steve: Yes, they will audit your own use and say, oh, you know, you ought to maybe fix this. It's like, ooh, thank you.

Leo: But the more I use it, the happier I am. It's just a great product.

Steve: Yeah, it's the right thing.

Leo: Yeah.

Steve: So Amazon has joined the team, or the league, of OAuth 2.0 site logon providers. For example, log on with your Facebook account. Log on with your Google account. Log on with your Twitter account. Now you can log on with Amazon. And when I heard that this morning, I immediately checked, and sure enough, they are - I was hoping they didn't just invent some random thing themselves because that would be dumb, and they didn't. They used OAuth 2.0, which is the technology we've covered in detail. We did a whole podcast on the way that technology works so that you're at some random site, you don't want to bother creating an account there because you just want to post something to a blog, or you want to buy something, but you don't want to go through all of the account creation headache. And they offer you the option, log on with Amazon.

And it's like, oh, I'm an Amazon user. So you just push that button. Either you're already logged into Amazon, or you bounce over to Amazon, log in with your Amazon credentials, and then the OAuth protocol working behind the scenes is able to provide an identity and whatever information the site has requested from Amazon. For example, they want - Amazon might do all of your purchase processing, but you want to provide your address and some information about yourself so you don't have to provide that all to the site. So that's now available from Amazon.

Leo: Oh, I just updated my Amazon password because that's probably a good thing to make sure that's secure now; right?

Steve: Yeah. And again, mark my words, every time on this podcast we have said something is going to happen, sometimes it takes a year or two, but then it happens. We are going to see an abuse of this. We are going to see sites saying, oh, log in with Amazon. And you're going to click the button, and you're going to be bounced to a fake Amazon site. That is - this is bound to happen. That is the problem with OAuth.

So when you're using any of this "log on with the high-value site credentials you already have," really look at where you're going. I mean, this is where you definitely want to see the extended validation coloration. Hopefully the site you're logging into offers that so you can verify you're at them. You want to look at the URL carefully to make sure that it's www.amazon.com, and it's not amazin.com or amazon.cat or something. I mean, look carefully because this is going to happen. And we don't want our listeners to get bitten by it.

Leo: I wish they would adopt two-factor. They probably won't because they're trying to sell stuff, right, and they don't want to put speed bumps in.

Steve: Well, they have two-factor now on a lot of their other services. Not on their - I don't know if it's on their main buy a book.

Leo: No, not that I know of, yeah.

Steve: But Amazon is supporting the proper OTP, the one-time password, two-factor, multifactor authentication for their S3 and web services and all kinds of stuff.

Leo: Oh, good. Oh, good.

Steve: So, yeah, they're definitely there. I wouldn't be at all surprised if they - they're probably just working it out, and then if they - I wouldn't be surprised if they move it forward. That would be good.

Leo: Yes, it would.

Steve: And Google announced something interesting, that actually what is, I mean, it just sort of seemed random to me. But then in looking at the details I thought, oh, there's some teachable bits in here for our listeners. And that is, Google has formally put out the word that they're going to be updating all of their SSL certificates. They figure it's time now to move from 1024-bit public keys to 2048-bit public keys. So they're doubling the public key length, which ridiculously exponentiates the security of the public key.

It's like, it's not - public keys, remember, don't scale their difficulty as quickly as much smaller symmetric keys do. That's why a smaller symmetric key, like an AES key, could be 128 bits or 256; whereas we need 1024 or 2048 bits in a public key. So the key length, I mean, if you had a key length like that for a symmetric key, well, there are no ciphers that operate on symmetric key lengths that long. You just don't need them. But you do in a public key. But still, this is great that Google is saying, okay, you know, these 1024-bit keys are seeming a little old.

But what was really interesting was that they were pre-announcing this to the developer community because it turns out many people have taken some shortcuts which could, if Google changed their certificates, could break authentication. But that's what the whole public key infrastructure, PKI, is designed to allow us to do. GRC, remember, I famously left VeriSign and went over to DigiCert, and I'm super happy for that. Nothing broke. And I upgraded my certificates to EV.

In fact, I just did that the other day for some other, some subdomains because I wanted to go 100% EV at GRC. Again, nothing broke. But it turns out that there are things developers could do that are essentially taking shortcuts. And apparently Google is aware of this and would like to say, look, if you've done these things, well, it's on you because we're going to give you some notice. So, for example, Google said, quoting them:

"For a smooth upgrade, client software that makes SSL connections to Google must" - and this was bold underlined italics must - "perform normal validation of the certificate chain." It's like, okay, well, who wouldn't? Well, we're about to find out. Also "include a properly extensive set of root certificates contained. We have an example set which should be sufficient for connecting to Google in our FAQ. Note: The contents of this list may change over time, so clients should have a way to update themselves as changes occur."

Okay, well, there they're just talking about the root certificates. So they're saying do normal validation, that is, don't do something screwy. And we'll talk about what that would be. Have a mature set of root certificates, meaning that Google's new certs are going to be signed by some major root certificate authority, like a DigiCert or a VeriSign or whomever. So you need those root certificates in order, you know, like a mature set. It may be that some people just have, for example, only have the one root that currently signs all of Google's certificates. And then, if Google were to change its provider of its certificate, then that would break software that was, like, highly Google specific. So Google is warning people, saying, eh, don't do that.

And then they also said: "Support Subject Alternative Names (SANs)." Subject alternative names are - that's the technical term for one certificate being able to provide coverage for multiple domains. For example, I have at GRC one certificate. And the formal name, the so-called "CN," the Common Name on the certificate is just GRC.com. Yet the certificate is valid also for www.grc.com. And had I anticipated it, I would have also put media.grc.com in there. And you can have many. And those are called "alternative names for the subject," so thus "subject alternative names." And the idea is that it's very practical to obviously have just one certificate that covers many domains, rather than having to have individual certificates per domain.

So this all exists in the very common certificate standard now, which is v3, Version 3, for this certificate standard, the public key infrastructure. And it's been around forever. But some clients may not know that SANs, the Subject Alternative Names, are really now important to support. For example, Google may not have been taking advantage of that in the past. Google is saying, they're broadcasting in advance, we intend to do that now. Everybody else has been doing it for two decades, so we'd like to, thank you very much. Please make sure that, if you've got any strange clients that are lagging way behind, they understand to look in the subject alternative name field if they see - because we may be providing a certificate whose common name doesn't match. And we're going to be relying on the client to accept, to look in the subject alternative name field for that match.

And they said: "Also, clients should support the Server Name Indication (SNI) extension," and I'll explain that in a second, "because clients may need to make an extra API call to set the hostname on an SSL connection. Any client unsure about SNI support can be tested against" - and then they give the URL - "https://googlemail.com. This URL should only validate if you are sending SNI."

Now, what SNI is, is another very cool extension which again has existed for quite a while, and adoption of it has been slow. The idea is that - and Leo, we've talked about this before. In a hosted website model, there's been a problem of each domain, each SSL needing to be associated with a single IP address. And the idea being that the client does a DNS query, for example, on, say, well, I'll use myself again as an example, GRC.com. That gives an IP address. And it then connects to the server at that IP and will only accept a certificate that says GRC.com. But as we know, there are hosting providers who have a huge number of websites all sharing a single IP.

Well, the way that works for web browsers is in the query header they have a host header that says this is the host that I'm wanting to connect to at this IP. And so that allows a multidomain hosting provider to hook that client up with the proper server because the client is saying I want this hostname at this IP. Well, what SNI does, Server Name Indication, that's similarly a way for the client in the initial SSL negotiation, in exactly the same way, for it to say this is the host I am hoping to connect to at this IP. Remember that this is important because we were talking about the client's query and the query headers, where that host header is. But that's in the actual dialogue after the

security connection is established.

So the point is that's too late for the SSL protocol to receive the information. It needs it in that first packet the client sends to the server, saying I'm wanting to negotiate a connection with this server. So Google is saying that they're going to be in the future, they have not been already, they're going to be relying in the future on clients saying this is the Google domain we're wanting to connect to at this IP so that the proper certificate - and the idea is that the server then has a bunch of certificates that all might be served at that one IP. And it chooses, the server chooses which certificate based on what the client has said it's asking for.

So again, these are all standards that have been around a long time. But, famously, software is slow to update, and it doesn't if it doesn't need to. And Google is saying, eh, you know, some of these things may break. Finally, they said, "On the flipside" of here's all the good things, they said, "here are some examples of" - remember we talked about, like, they said you must properly validate the certificate. And it's like, well, okay. Who wouldn't? Well, apparently people don't. So they say, "Here are some examples of improper validation practices that could very well lead to the inability of client software to connect to Google using SSL after this upgrade." And so the first one is, they said, for example, do not do this. And that's "matching the leaf certificate exactly, e.g. hashing it."

So, okay, now, get this. Apparently, as we know, the right way - by "leaf certificate" they mean the server certificate, the last one in the chain. Apparently there are some clients which know what the Google certificate is, and they just have a hash for it. And so they don't do any public key chain following. They don't do what you should, which is take the Google certificate and then build a chain back to a valid root certificate that you have in your local root store, and then make sure that every certificate in the chain has signed the successive one so that you trust every linkage. And since you trust the root, you trust the leaf.

Well, apparently not everyone does that. They just say, oh, look, this is Google's certificate. We're just going to remember that. And Google is saying, obviously, when we change our certificates, the hashes of those certificates are going to change, and so suddenly you're not going to be able to make a secure connection because you were, like, doing some lazy ass way of SSL. So my opinion is you deserve what you get, if that's what you're doing.

And then they say also, "matching any other certificate, even a root or an intermediate signing certificate exactly," in the same fashion. I mean, yes, I mean, nobody should be doing that. But apparently Google believes some people are. Or "hard-coding the expected root certificate, especially in firmware." Google says: "This is sometimes done based on assumptions like the following: the root certificate of our chain will not change on short notice." And the fact that it never has doesn't mean that it's not going to. And apparently Google plans on that happening. Or making an assumption: "Google will always use Thawte as its root CA. Google will always use Equifax as its root CA. Google will always use one of a small number of root CAs." Google is saying don't make any of these assumptions. We're planning to break all of these.

Leo: We're wily.

Steve: Or "the certificate will always contain exactly the expected hostname in the common name field, and therefore clients do not need to worry about SANs." So, and then finally, "the certificate will always contain exactly the expected hostname in a SAN,

and therefore clients don't need to worry about wildcards."

So basically Google is saying, look, we're aware that shortcuts have been taken, and we're about to break all of those. So get your acts together. If you want to be able to establish secure connections to us, just do it right, 100% right. Basically what they just gave us was all the things not to do and the things you need to be able to support. And all contemporary SSL systems have been doing this for years, for decades. So it must be weird little boxes in the closet, you know.

Leo: Yeah, as always.

Steve: We often talk about boxes in the closet. And Google is saying, you know, we need the freedom to change. And in fact Google maybe sort of boxed themselves in a little bit by not having changed before. Like they are rigorous with the way their decisions have been made. If they always were using the same root CA, then someone could go, eh, well, you know, we'll check the chain, but we're only going to import the root CA that Google uses because that's Google's root. And Google is saying, eh, no. I can't wait to see who they switch to. That'll be really interesting just to know, to wonder what their logic has been.

Leo: They probably have a randomizing Python script that just picks one.

Steve: Yeah, I imagine they'll be dealing, I mean, they're going to certainly care about security. They're going to want to get a really good provider. It'll be fun to see.

Leo: Do you think they have, like, a lot of certificates?

Steve: What they have is special. They have a certificate that lets them make their own. And that's special.

Leo: Ah.

Steve: So if you look at a Google chain, they are signed, I think, by Equifax. But they have a signing certificate from Equifax. So Google is minting their own certs at will for all of their various domains and subdomains. And, I mean, they're a global network. It's much easier, and actually it's safer, to use multiple certificates scattered around because if one ever did get compromised, the rest of their network would not be compromised. If they used one certificate globally, oh, goodness, I mean, if that ever got compromised, it would take the whole world down, their whole world.

Leo: And then Larry keeps that certificate-maker in his pocket so that no one can get it.

Steve: Believe me, it has got to be guarded. There's probably some serious technology surrounding that.

Leo: We'd love to see that. They're never going to say because they don't want anybody to know.

Steve: Yeah. But it's - ultimately they - so they're being - their certificate is signed by a major root, like Equifax. And but it uniquely has its own signing capabilities. So Google can mint their own certs for whatever duration and reasons they wish. And they have that privilege because they're Google and trusted.

Leo: I want to meet Gozer, the Keymaster, the guy who holds that cert. There's this one guy, and you have to go to him and say, hey, can I come over at 3:00 o'clock this afternoon? I need to mint some certs.

Steve: It would be a...

Leo: It's a perfect breath mint.

Steve: Be nice to have. So I tweeted this yesterday just because I thought it was very cool. And I wanted to give a shout-out about it to our listeners. It's called - it's on Kickstarter. It's a Kickstarter project, lots of time to get in if you were interested [kck.st/154D8Ba]. It's called Linkbot, L-i-n-k-b-o-t. And so the URL is linkbot-create-with-robots. It's just a very cool project. They come up with a modular robot module, essentially. It's a common block with Bluetooth. It's rechargeable. It's got blinky lights. It's got a beeper so it can make sound. I think it's got two motors that operate independently.

And the idea is that you can stick these things together and make robot arms and make caterpillars and things that walk and do stuff, all just sort of in an erector set kind of mode. It's Arduino compatible. It's got a cool feature where you can, like, if it was articulated, if you had like a multi-joint thing, you can move it to a position and then say, okay, remember this, and then move it again and then remember this, and move it again and remember this. And then it'll go back and follow that whole sequence for you without any programming. And it's all open source. There is a programming language that it's tied into. Anyway, it's just very cool.

So I wanted - I did tweet about it yesterday. For those who follow me, you already know. But Linkbot, Create With Robots over on Kickstarter just came across my radar, and I wanted to share it with people. And I have a neat story from a friend of the podcast who has written before, Robert Osorio, who's in Lady Lake, Florida. He wrote, he said, "SpinRite Resurrects Dying SSD." This is not something we've heard about before. And actually the way this SSD is dying is frightening. And it's interesting that it even can die in this way. So listen to this one.

He said, "Steve, the other day I got a really nice example of SSD resurrection by SpinRite. A client gave me a year-old, so not very old, Dell XPS 17 with an SSD boot drive that took seven-plus minutes to boot. And when it finally did boot, it ran horribly slow, with long pauses at the desktop. The Windows Event Log showed multiple disk errors, and the Dell hard drive diagnostic confirmed the drive was bad and gave me an error code. And since it was still under warranty, I had a replacement coming via overnight shipping.

"However, I really wanted to image the drive, if possible, since there was a lot of specialized software on this laptop, and it was a mission-critical system. A fresh OS install would have been a long and tedious ordeal in this case. Needless to say, all attempts at imaging the drive had failed when it encountered errors. I ran SpinRite Level 2 scan on the drive, and it detected around 12 unrecoverable bad sectors.

"After running the scan, the drive booted instantly into Windows. It also allowed me to create a drive image using an Acronis Boot CD, which I subsequently recovered to the replacement drive, so all was well. To show how bad a condition the drive was in, I ran a Level 2 scan three more times just for fun, and each time SpinRite detected even more unrecoverable sectors in different locations. Thanks for a great product. Bob."

So this is interesting because this says that - remember that the cells in an SSD, the bit cells are - they're so-called MLC, multilevel cells, in these consumer large SSDs. So essentially it's a capacitor which stores an analog voltage. If it's a two-bit cell, it'll store four different voltage values, which means it could be zero, it could be one third, two thirds, or full.

And so the problem with multilevel cells as opposed to single-level cells is you have three thresholds where you need to detect the voltages, rather than just one. In a single-level cell, you'll either have a zero or a one, zero voltage or full voltage. And then you have a - technically, typically, half of that is the threshold. If it's below that, then you know you got a zero. If it's above that, you've got a one. Because the idea is this will, over time, these capacitors tend to discharge. And they last a long time, but that's the technology is just charge stored in a capacitor. And so the multilevel...

Leo: I had no idea it was that simple.

Steve: Yeah. And so the multilevel cells, the reason you would store, obviously, four voltages is to get two bits per cell. That instantly doubles the density of your SSD. You can go from a 128GB to a 256GB. So that's really valuable. But the problem is you need to store two bits, meaning four values. And now you've got three threshold levels. Is the voltage above two thirds? Is it below two thirds but above one third? Is it below one third but not zero, and so forth.

So what is happening on this particular one, I mean, this is - it's good that he's - this is definitely defective. It is dying as we watch. I mean, remember that SpinRite is running through and reading the value from the cell. If it's enough uncorrectable, if it requires enough correction, then that sector will be swapped out, and another different one will be put in. But these things are dying, like, hour to hour. So he didn't wait too long getting - running SpinRite on this and then immediately doing an image in order to get this moved over to a new drive. And it's really interesting to see an SSD that is in this bad shape after apparently only a year.

Leo: I don't think that's common. And most of the drives nowadays are MLC. It's hard to find SLC SSDs.

Steve: Yeah, that's all I buy is SLC because of course I'm the SpinRite guy.

Leo: Where do you - is that the Intel? Where do you get that?

Steve: Mine, I bought OWC, a beautiful...

Leo: Well, I did, too, but it's an MLC card that I got. I got the Accelsior.

Steve: Well, mine were the individual drives. And I've looked - but they're still small, also. I mean, I didn't get multi terabyte. I'm down, because, remember, I'm in assembly language, so none of my stuff is big.

Leo: You don't need a lot of space, yeah.

Steve: Right. So I think these are, like, 60GB drives. And I'm running them in RAID 6, so I'm only getting 40GB. But that's way more than I need for my server because the server runs on, like, 2GB.

Leo: But I suspect that a lot of the issues with MLC have been solved because it really is hard to get SLC. But I don't know. Even some of the...

Steve: Well, high-end...

Leo: I got a pretty high-end card that's - it's MLC. From Other World Computing, yeah.

Steve: Yeah. For what it's worth...

Leo: Oh, you get the Enterprise, probably; right? That's what you're getting.

Steve: Yes, they are the Enterprise...

Leo: Yeah, that's why they're MLC. SLC.

Steve: Way, way more expensive.

Leo: Yeah, yeah, sure is. Never mind. We're talking 50GB for $300, $6 a gig.

Steve: Yeah, I paid that.

Leo: Yeah, yeah. And for that you get SLC.

Steve: Yeah, Enterprise class. And then I put in a RAID 6, so I got 100% redundancy. Any two of those can die, and I'm still just fine, so, yeah.

Leo: That's Steve, right there.

Steve: But I don't have to worry about my server anymore. That's all taken care of.

Leo: Yeah, that's right. Oh, that was for your server. Well, that makes sense, yeah.

Steve: Oh, yeah, yeah, that's for the GRC, that's my...

Leo: Oh, no, that makes a lot of sense. You wouldn't want to do that necessarily on a computer, you know, a laptop.

Steve: That would be overkill.

Leo: Yeah. So CueCat is an IM, instant messaging solution? Is that right?

Steve: Okay, so, yeah. Let's frame this properly because I started off putting too much emphasis on Cryptocat. Cryptocat is definitely fun, definitely worthwhile. It's a web browser plugin and also available apparently as a standalone Mac app. You have to have 10.6.6 or later. But it's also available as a plugin for - and I'm trying to look through my notes. Where is it? There it is.

Leo: Chrome.

Steve: Chrome, Firefox, and Safari. And I did install it on Firefox to play with it. It's a cute little tiny thing, 635K. And then you have to restart Firefox, and then you're there. So Cryptocat is a browser plugin-based secure chat system. And now we're going to spend the rest of this podcast explaining what secure chat is because it's different than, for example, PGP-style secure email. But so what I like about Cryptocat is that it lets - it's, like, almost no installation, and bang, there you are able to set up a chat relationship with anybody else in the world, and it is utterly secure.

It is - there is - okay. And I have to define what "secure" means because, again, I'm using secure chat in a way that I haven't defined yet. But what I like about - the reason Cryptocat is fun is it's just zero friction to get into it. You load a plugin, and bang, you're going. So if there's anybody you want to do chat with, you can.

On the other hand, it turns out the underlying protocol, which is what I find interesting, I mean, the client there is very nice. The underlying protocol is where the meat is, and this

is where the use of the word "secure" comes into play. And the protocol is widely available. It's all over the place. It's been around some - Ian Goldberg is one of the coauthors of the protocol. He's been in math for a long time. He's now a cryptographer. He was - we once mentioned him because he's the guy that realized that Netscape Navigator's random number generator for its SSL connections was no good, and that completely crippled the security of Netscape Navigator's early SSL. That was Ian who figured this out in his early 20s. But he's delivered some papers about this from at least dating back to '05.

So this OTR, that's the name of the protocol, Off The Record, and as I mentioned at the top of the show, it is already widely available, so you can use it in existing instant messaging clients. You would obviously need to use it at each end. And in fact it refuses not to have the other end being used if it's being used. The site where all of this can be found, and links, is Ian's site, which is Cypherpunks.ca, C-y-p-h-e-r-p-u-n-k-s dot ca, Cypherpunks.ca. And /otr, sorry, slash Off The Record. I think if you just go to Cypherpunks.ca, it tells you you need login credentials, and so you want to go to /otr, which is a non-protected directory on that server.

So Ian I looked at, like, so who is this guy. For three years in high school, so for I guess his last three years of high school, he was a member of Canada's team at the International Math Olympiad where successively he brought home a bronze, a silver, and a gold medal. He got his bachelor's in pure math from University of Waterloo in '95 when he was 22 years old, and that's when he uncovered the crucial flaw in Netscape Navigator's SSL random number generator. This is the random numbers that protect the choice of key when you're negotiating your symmetric cipher, and we all know that crypto needs really high-quality random number generators. It turns out that Netscape Navigator's wasn't very good. We've talked about that on podcasts in the past. This was Ian who found this. Then he went on to get his PhD at Berkeley with a thesis titled "A Pseudonymous Communications Infrastructure for the Internet." Today he's back in Waterloo as an associate professor and university research chair, and teaches some courses every quarter in security and privacy.

So, okay. So what is secure chat, and how does that differ from, for example, secure email? What's the problem with secure email? Turns out there are some characteristics of secure email which you may not want in chat. For example, the way - we're familiar with PGP. And in general we're familiar with the typical secure crypto technology. Often in crypto protocols we talk about actors Alice and Bob, you know, A and B. And so Alice and Bob are our two guys. And so, for example, Alice and Bob each have public key pairs. And we're talking about the normal PGP model, the traditional protect your email communications. So they generate a public key pair. They keep their private keys private, and they make their public keys widely known.

And Alice wants to send Bob a message. So she uses her private key to sign the hash of her message. And since only she has her private key, then only she is able to sign the hash of her message. So then, after signing it, she knows she's going to send it to Bob, and Bob has published his public key. So she uses his public key to encrypt the key that she uses to encrypt her message and sends that off to Bob. Well, only Bob knows his private key. So he's the only person who's able to decrypt the key that Alice used to encrypt the message. So he uses his private key, decrypts the key that Alice used, then decrypts the message.

Now he wants to authenticate that it's from Alice. So she signed the message with her private key, which only she knows. So he verifies the signature using her public key. And if it verifies, he knows that only the matching private key could have been used to sign it. So that's our standard, here's how we authenticate and we encrypt, that is, we get

privacy in something like PGP. That's the technology. And so, like, okay, that's everything we want; right? Well, maybe. But we can do better than that.

And that's what these guys recognized when they created this notion of a secure chat protocol because, what if Bob's computer was grabbed by a bad guys? Maybe by criminals, by competitors, or I don't want to call the FBI "bad guys," but the point being that these two people, Alice and Bob, want to keep their communications private for whatever reason. So if the computer were subpoenaed by a three-letter initial organization or grabbed under the Patriot Act...

Leo: The NFL?

Steve: [Laughing] Right.

Leo: CIA. NSA. FBI.

Steve: NSA, CIA, FBI, one of those guys.

Leo: Yes.

Steve: Then the point is that all of Bob's keying material can be recovered, and then they're able to do everything that Bob was originally able to do. And that's the key, originally able to do. So that means that they can decrypt all of his past messages to learn their content. So there goes their privacy. And they can also learn that Alice provably sent them because the same strength of authentication that operated at the time the message was received persists. And that's key. So they not only know what Alice sent, but they can prove she sent it because Alice's key is public, and only somebody who had the private key could have signed those messages. And that's absolute mathematical certainty. So this is a problem. We can do better.

So imagine just a chat, a conversation in a quiet room. Alice and Bob talk. No one else can hear them. So they have privacy. No one else knows what they say. No one can prove what was said, not even Alice and Bob, after the fact. Their conversation itself is ephemeral. It's happening, and they're each present, nobody else is, and they can assert whatever they want to, but that's not proof. So we say that such conversations are "off the record." I mean, there's even legal standing for an off-the-record conversation. You have an off-the-record conversation with a reporter, and that means something legally. He's like, you're not recording this, this is off the record, not for attribution, blah blah blah.

So how do we achieve that, a different set of goals than, for example, secure email? We want a secure chat. So what does that mean? That means a technology we've discussed before known as Perfect Forward Security. That's the term used where no subsequent disclosure of cryptographic keys can disclose anything that was previously encrypted under those keys. So that's very cool. That's not what PGP has. PGP does not give you perfect forward security. As we said, if Bob's keys got loose, or Alice's keys got loose, their conversations are decryptable.

So we need something which is ephemeral, something which is not going to last, on

purpose. It's there long enough for the conversation, but inherently not long enough for a record. But the other thing which these guys have done which is very cool is deliberate repudiable authentication. You want to be able to repudiate the dialogue, meaning that you want to be able to say, "I didn't say that," and for there to be no way for anyone to prove you did. Which, again, that's what a conversation between two people in a quiet room with no eavesdropping gives you. Even the other person, the other party in the communication can allege that you said something.

Leo: I repudiate you.

Steve: You can deny it. You can repudiate that assertion. So we're not talking about this casually, though. We want this in, like, technology. Okay. So Perfect Forward Security means, technically, that a future key compromise must not reveal past communication contents. So we do that by using so-called "ephemeral" or short-lived keys. And what's cool about this dialogue, that is, this protocol, is they are constantly negotiating new keys and discarding the keys they have just used.

It turns out that what they use is a Diffie-Hellman key exchange, which we've also discussed in the podcast in the past. That's the cool concept where it doesn't matter which sequence you exponentiate in. They each create random numbers. They raise a prime number to this random number power. They take the result mod another prime number, keeping just the residue, just the remainder, essentially. That they send to the other person. And the other person does the same thing and sends their residue to the other person.

So they exchange their residues. Then they each do the same thing again. And they arrive at an identical result. Yet nobody monitoring the conversation, nobody intercepting this exchange of residue can recreate this common number they got because each of them keeps their own secret. They only exchange the residue from the use of that secret. And that, because of taking something to a high prime power and then taking the modulus to another prime, all you get is the remainder. That's one of these hard problems. You can't go backwards. Like a hash that throws away lots of information and distills it, this is a different kind of distillation process. But it's very fast to do.

So this Diffie-Hellman exchange you can do per message on the fly, which is exactly what these guys do. Every time, as you're chatting back and forth in the chatroom, each side is continually generating another Diffie-Hellman exchange key and sending it along with the chat payload to the other side. They're always keeping the current and the next key present. But they're always retiring and, like, deliberately deleting, they're overwriting in memory and deleting the previous key. So they're constantly retiring the previous negotiated keys.

Also, there is a public key technology here because the one thing the Diffie-Hellman key exchange doesn't do is authenticate. Notice that, if there was a man in the middle, he could exchange keys with Alice and then exchange - or exchange mod residue with Alice and turn around and exchange modulus residue with Bob and set himself up as, you know, pretending to be each other for the other and insert himself in the middle.

So you do use, in this protocol, and it's spelled out, there is a public key component where they use their public key only for authentication, not for signing. And that's the key. The mistake, such as it is, that PGP makes is that Alice signed her message using her private key, so it was non-repudiable. Someone could say this came - this was signed by Alice a month ago. Doesn't matter that it was a month ago. She signed it. So, and it's

because she's using her private key of her public key pair to do the signing. This protocol specifically does not do that. What this protocol does is sort of interesting.

Once you've got this key agreed, they hash that to get the encryption key, and then they hash that again, that is, they hash the encryption key to get, so it's derivative, to get a keyed hash. We've talked about keyed hash, so-called "message authentication codes," MACs. The idea is that a message authentication code is a keyed hash which is like a digital signature that is also based on a key.

So since both sides have - they've used Diffie-Hellman to exchange and get a big key. They then hash that to get the encryption key, which they'll encrypt their messages to each other with. And remember that they are, on the fly, they're discarding these. So those do not persist. Which means that no future compromise of anything they do, even other conversations with other people or the same people, that will never allow previous conversations to be decrypted because this is per event keying, essentially.

But how do you get authentication? How do you know this is really from each other? You've got - I would say, well, the fact that you can decrypt it and get something meaningful, only somebody who had the key could encrypt it, but that's not good enough for cryptographers. They want an actual authentication. So you hash the cipher key to get this so-called - the MAC key. And they each do that. So they each have the same MAC key, the message authentication key. And so they simply hash their message with the same message authenticate key, and append the result to the message, which they then encrypt and send to each other so they can provably authenticate.

But notice what happens. Since they're using the same hash, they're not using each other's signature. They're deliberately using the same hash. That's exactly like Bob making something up that he says Alice said. But if it's in this communication, he had the hash. And that's all you - and he had the cipher key, meaning he has everything necessary to forge a message from Alice. Meaning that you can demonstrate mathematically that, unlike PGP, where only the other person can be shown to be the provable creator, this Off The Record protocol creates identical cipher material on each side, making them equal in the conversation. Meaning that there is no way for Bob to ever assert that Alice sent him something because he has an equal ability to create the same message that Alice sent.

And then they go - these guys go one step further, and that is, as they are retiring the cipher keys, they're always creating new ones. And as soon as they receive a message under the new key, they overwrite and discard the previous one. They also send, they deliberately send the previous message's hash key as part of the following message, just to sort of put it out there in the public, to sort of further say anybody who wants to can forge the previous message. They can't forge the current one. They can forge the immediate previous one. And so they attach that authentication key just to sort of thumb their nose at anyone trying to make a case from the data which has been exchanged by deliberately giving away everything necessary to forge the previous message. And that's Off The Record protocol. Simple.

Leo: Yeah. While you were doing this we had a kind of a back-channel chat going, and I started it in Chrome, and I exited the Chrome chat. The chat persisted and then went into the Mac app, and everybody's still in there. And it's pretty nice.

Steve: Yeah, well, so that's an implementation which I can recommend. The guys that are doing it have gone through a full audit. They had a full code audit. All of their code is

open source, available for people to look at. It's also multi-language. They've got, I don't remember, it's like 30-some-plus languages. Even Esperanto. It's like, what? Who actually speaks Esperanto? I guess there actually is a community of...

Leo: I don't think there's, like, native Esperanto speakers.

Steve: No, well, apparently there are people who are...

Leo: There are people who embrace it.

Steve: Okay, they're well-embraced Esperantans, yes. So, yes, you can use Esperanto, if you wish.

Leo: Awesome.

Steve: For Cryptocat. Anyway, so I know our audience. There will be people who love the idea of being able to use either existing chat clients, for example, when I get into the TWiT chatroom, I use XChat as my little Windows chat client.

Leo: Love it, yeah, yeah.

Steve: Yeah. XChat supports this protocol. XChat supports Off The Record protocol.

Leo: Oh, that's neat.

Steve: And so you could use XChat. And also there are a number of Jabber clients. And so this is Jabber compatible. Is there something called Pidgin?

Leo: There's Pidgin, which is a multi-client or multi-protocol client for Windows. And, I think, for Linux.

Steve: And this is also, it is a Pidgin plugin. There is a plugin for Pidgin.

Leo: Irssi supports it, as well. There are quite a few chat sites.

Steve: Yes.

Leo: Yeah, that's good.

Steve: Yes. So anyway, so now, I mean, this is the technology. The technology is nailed. And as I said, I know our listeners. Among our population here are people who just, I mean, actually there's something cool about just, even though you don't need it, just knowing that this thing is so insanely bulletproof that nobody can eavesdrop on it, unlike Skype Chat, we know, which Microsoft is logging in the clear and has the ability to send us the history on a new machine that is set up, so obviously they have our chat. Just the idea that you could have a point-to-point conversation with absolute security in a browser plugin, as we've said, Chrome, Firefox, and Safari. So that's it.

Leo: It's neat, yeah.

Steve: Yeah, they nailed it. They really got it.

Leo: Yeah. So you can download it, but you could also use it as a plugin in existing chat clients.

Steve: Yup. And again, it's Cypherpunks.ca/otr, for Off The Record. In fact, I think in the PDF I sent you I called it OTP because, of course, One-Time Passwords. But no, OTR, Off The Record. We're getting our acronyms confused.

Leo: Yeah, the Cypherpunks have been around for a long time and actually are a really great group.

Steve: Yes, and this Ian and his gang who have put this together.

Leo: Well, that's neat. Be nice if you had a plugin for Skype, but you don't. So you just have to, if you want private chat, use something else.

Steve: Yeah, I think Skype, I mean, now I'm seeing commercials on, like, ads on Skype.

Leo: Well, Microsoft is really pushing it heavily. And, you know, but we use IRC, which is completely in the clear and open, and anything we do on our chatroom is logged and visible, and you would never expect it to be private; right?

Steve: No. Also I should mention that the Cryptocat folks are very close to releasing mobile clients. So there will be iOS and Android versions of Cryptocat also available.

Leo: You know what would be really interesting is to see Google support it with their new Hangouts chat. I bet you there's a Jabber plugin for it.

Steve: Yes, yes, there is Jabber support.

Leo: Be really nice to see Google do that. And if you could flip a switch in Google Hangouts to do that, boy, that would really change the world.

Steve: Bring down the cone of silence.

Leo: Yeah. Just say, yeah, call it the Cone of Silence. That's good. Steve Gibson does this show every Wednesday, 11:00 a.m. Pacific, 2:00 p.m. Eastern time, 19:00 UTC on TWiT.tv. We want you to watch live if you can, always fun. Then you could try out things like the Cryptocat chat. We were - there were about 15 people saw me start it up, joined us in there.

Steve: Oh, cool.

Leo: It was kind of fun, yeah. But you can also get on-demand audio and video after the fact. Steve hosts the smallest version, the 16Kb audio version on his site, GRC.com. Right next to that you'll find written-word transcripts, written by a human being, Elaine Farris, so they're very good. And so that's a great way to read along as you listen along.

While you're at GRC.com, don't forget to try SpinRite, world's finest hard drive maintenance and recovery utility, and all the freebies Steve offers. This would be a good time to visit the Password Haystacks page, among others. And if you have a question about anything you heard today or other shows or just something you just read online, like that Ars Technica article, Security Now! feedback is GRC.com/feedback. And we'll answer 10 questions next week.

Steve: Yeah.

Leo: If the world of security permits. If there's no breaking news. So Security Now! feedback is at GRC, Gibson Research Corporation, GRC.com/feedback. Steve's on Twitter as @SGgrc. And what else? I guess that's about it. If you want full quality audio or video, we've got that at TWiT.tv/sn.

Steve: When you went to Cryptocat, did you just go into the lobby as a chatroom?

Leo: There isn't a lobby. What happens, you know, I downloaded the Chrome plugin, but I also installed the Mac plugin. If you want to see, I'll show it to you.

Steve: Well, there actually is a lobby. If you...

Leo: Ah. Well, see, now, that raises an issue because, if you know the name of the chat, you can enter the chat.

Steve: Yes. And but they also, if you click, there's a field - you have to poke around there. But you're able to get your own information.

Leo: Yes.

Steve: And you can click a button.

Leo: You can offer that to somebody, say let's do a private chat.

Steve: Exactly. But it's fun because they have, and I didn't mention this, there is a multi-party variant of this. And so they've extended this Off The Record protocol, which is inherently a point-to-point two-party exchange, there is a multi-party variant, which they're working on. And that's what they're using in this chatroom where you can go if you just use lobby. If you hover your mouse over...

Leo: So lobby can be a chatroom, as well. So I'm in the lobby now.

Steve: Yeah, it's sort of a default. I was just going to say, though, if our listeners wanted to chat with other of our listeners, it's very quiet. I've been logged in there for a day and just, like, one or two people come in and don't say anything, and they leave.

Leo: But that's where you would find somebody and then create a private one-on-one chat through that.

Steve: Yeah.

Leo: Is that the idea?

Steve: Yeah, well, or you'd probably - somehow you do need to exchange your keying material off-channel.

Leo: Ah. So I've just posted my fingerprint, which is visible in the client, for private conversations, and then a group conversation fingerprint. That's the one I guess we're in.

Steve: Exactly.

Leo: So I'd somehow supply him with this number, this key.

Steve: And that would allow him to find you.

Leo: We'd have a private chat online. Yeah, no, it's very cool. And then once you're in the lobby, of course, you can chat with somebody else, but you can also leave the lobby and go to another chat. We had a Security Now! chat. And they were fun. There are people in there. It looks just like IRC. Not, you know, I mean, it's kind of rudimentary. You can't control the font or the colors, although there's a - I found a CSS file and a resources package. You could probably modify it and hack around.

Steve: Mostly it's super secure, so that's so cool.

Leo: Yeah, it's nice.

Steve: Yeah.

Leo: Thank you, Steve Gibson. We'll see you next week.

Steve: Thanks, Leo.