EPISODE 445

[INTRODUCTION]

[0:00:01.1] JM: Public key encryption allows for encrypted private messages. A message sent from Bob to Alice gets encrypted using Alice's public key. Public key encryption also allows for signed messages, so that when Alice signs a message, Alice uses her private key and Bob can verify it if Bob has her public key.

In both cases, in both the case of sending an encrypted private message and signing a message, Bob needs Alice's public key. If Bob gets that public key from an e-mail message, Bob is trusting the e-mail message is secure. If Bob can't ever verify that very first message that contains the public key, he has no way to verify the messages that come after it.

This is the problem of key distribution. How do I get your public key securely? Key distribution undermines the usability of PGP encryption. Serious encryption advocates will sometimes meet in person to exchange pieces of paper containing public keys.

Keybase is a company that attempts to solve the problem of key distribution by having users connect social media accounts and devices to Keybase, in order to collectively verify who you are and then give you the power to share your public key reliably.

Max Krohn is a founder of Keybase, and he was previously a founder of SparkNotes and OkCupic. He was on the show a few years ago to discuss the basics of Keybase. In this episode, he explores some of the abstractions that Keybase has built on top of its core identity tool. That includes Keybase file system, Keybase Teams and Keybase Git.

We do break down the basics of Keybase, but if you want a more thorough explanation, you might like to check out that older episode and you can find the first episode I did with Max on the Software Engineering Daily app, which is on iOS and Android. You can find all of our old episodes in addition to that episode with Max.

With that said, I hope you enjoy this episode with Max Krohn.

[SPONSOR MESSAGE]

[0:02:14.1] JM: DigitalOcean Spaces gives you simple object storage with a beautiful user interface. You need an easy way to host objects like images and videos. Your users need to upload objects like PDFs and music files. DigitalOcean built spaces, because every application uses object storage. Spaces simplifies object storage with automatic scalability, reliability and low cost. But the user interface takes it over the top.

I've built a lot of web applications and I always use some kind of object storage. The other object storage dashboards that I've used are confusing, they're painful, and they feel like they were built 10 years ago. DigitalOcean Spaces is modern object storage with a modern UI that you will love to use. It's like the UI for Dropbox, but with the pricing of a raw object storage. I almost want to use it like a consumer product.

To try DigitalOcean Spaces, go to do.co/sedaily and get two months of spaces plus a $10 credit to use on any other DigitalOcean products. You get this credit, even if you have been with DigitalOcean for a while. You could spend it on spaces or you could spend it on anything else in DigitalOcean. It's a nice added bonus just for trying out spaces.

The pricing is simple. $5 per month, which includes 250 gigabytes of storage and 1 terabyte of outbound bandwidth. There are no cost per request and additional storage is priced at the lowest rate available. Just a cent per gigabyte transferred and 2 cents per gigabyte stored. There won't be any surprises on your bill.

DigitalOcean simplifies the Cloud. They look for every opportunity to remove friction from a developer's experience. I'm already using DigitalOcean Spaces to host music and video files for a product that I'm building, and I love it. I think you will too. Check it out at do.co/sedaily and get that free $10 credit in addition to two months of spaces for free. That's do.co/sedaily.

[INTERVIEW]

[0:04:33.5] JM: Max Krohn is the co-founder of Keybase. Max, welcome back to Software Engineering Daily.

[0:04:38.8] MK: Thanks, Jeff. Great to be here.

[0:04:40.6] JM: We had an episode a couple years ago where we discussed the foundational concepts of Keybase. But let's reintroduce people who may not have heard that episode. How do you describe Keybase as a company today?

[0:04:54.4] MK: Sure thing. We are a company that is trying to make it easy for people to use public key technology in their lives. That means, maybe three different things. The first is that we make it easy for people to publicize their cryptographic identity, which means that we make it easy for people to have a public profile that shows how you might know them through social media, and also what their public credentials are so you can communicate with them securely, or if you want, so you can verify things that they've authored.

Those two things are like the Yin and the Yang of public key crypto. It's both useful for encryption and also for signing and verification. To get back to people's public profiles, if you look at my public profile on Keybase, you'll see that I have verified ownership of several social media accounts, like a Twitter account, a GitHub account or Reddit account and etc., and also that I'm the TNS administrator of several DNS domains.

What that means is I've actually signed statements in my Keybase profile saying that I own control of these resources, and then if other people who want to check that out can see that I've actually posted those on the resources that I claim to own. Therefore, anyone who would be trying to impersonate me would have to go through all those steps, which might be very hard, because they'd have to break into all of my different social media accounts and post bogus things on.

That's the first thing that the Keybase really is for people. It's a way for them to publicize who they are and the cryptographic keys that correspond to their to their identity. That's where we really started, but we realized quite quickly that in order to give people that set of features, we had to provide them a convenient way to manage their private keys. For every public key you have, there's obviously a corresponding private key that only you should hold and no one else should have access to.

Traditionally, it's been very hard for people to manage private keys. If you ask PTP how to do it, they'll tell you it should be on a USB thumb drive that you keep in a safe somewhere, and you're probably on your own if I'm going to move it onto your smartphone, because obviously you shouldn't moving the secret key in a way where any sort of bad guy might have access to, and that's very hard to smartphones.

What Keybase also provides is a way for users who don't have to have the most tactical know-how in the world to manage their private keys. The way we do that is that every device you install a Keybase on gets its own private key. If you want to add a new device, that means you have to introduce it to one of your old devices and then they share authorization with each other, so now your new device has a new private key that is unknown to any of the other private keys, and also – sorry, any other devices and also unknown Keybase. It's unknown to anyone, but the owner of that device.

We've tried to make that very easy for people, because it's a tricky thing to get right. We also want to get people a way to recover their identity and not get totally broken into if they manage to lose one of their devices, which actually as you can guess happens a lot. If your hard drive crashes, if you leave your phone in a cab, that should not be the end of your cryptographic identity.

You should not have to reset. Rather, you should be able to say with front of your other devices that, "Oh, hey. I lost my phone in the cab, so just don't use that device anymore. Just use the other three devices that I still have access to." Then you can go on and not have to reset everything about you publicly, which is also a pretty key innovation over things that used to exist.

Those are the two things that Keybase is really trying to do at a technical level to make it easier for people to use public key. Maybe one level up from that is we want to give people applications that they could use on top of this public key technology that are actually useful in their lives, so people find it very useful to send chat messages, to share files and recently to share get repositories. We want to make it very simple for people to use the public key infrastructure of Keybase, along with these applications that they're currently using today just in a less secure format.

[0:09:04.4] JM: Public key encryption is widely used when all of my web traffic and my chat messages, they go through HTTPS already on the internet. What used cases are you solving for?

[0:09:17.8] MK: That's a great point. It's true that if you're a bank, or if you're an internet service, or maybe if you're a technically sophisticated website hoster that you can go and get a public key and a corresponding private key for your website. That means that anyone who's at Starbucks who wants to contact your website or contact the bank, they'll know that Starbucks can't see the messages in between.

That's obviously great for a lot of people's usages and clearly, now that we know that WPA2 is pretty broken, that just came out today. It's good that people are using this technology. However, what that means is that whenever you communicate with your bank, let's say you use that example, fine Starbucks can't see what you're talking about, but the bank sees everything you're talking about as you wanted to.

Maybe a better example is like, let's say you're communicating with Dropbox. You're syncing files to and from Dropbox's servers at Starbucks and Starbucks can't see your files, but Dropbox can see your files. So Dropbox is able to index them and back them up and do whatever sort of internal data analysis they want to do on your files. That might not be a good idea for a lot of different people.

One thing you might be afraid of in that case is that there is someone evil within the organization who is snipping on people's documents and wants to figure out maybe the cap table of your company if you decide to store it on a Dropbox, or even if you give these companies the benefit of the doubt, which personally speaking I do. It still is a question that they might get hacked to some point. When they get hacked, what that will mean is anything on their databases will become open for anyone to download, and all of your secret documents will become public.

We've seen it happen with a number of different large scale providers, like Equifax and Yahoo and some of these other providers have better track records, but they're just one mistake away from letting the cat out of the bag, or the horse out of the barn as where – If you're in a relationship with these providers where you can lose your password and they can just reset your account, then you know that everything they have is somehow available to them in plain text.

That's true of almost everything people use on the internet today. In 2017 where security seems to be getting worse all the time, it seems like people might want to take matters into their own hands. The only thing you could do as a user who's slightly distrustful of your service provider like Dropbox or Google Drive, is to make sure that they only get to see encrypted data. That way if their data is ever hacked, all the hackers and everyone else gets to see is the encrypted version of your data and not the data itself.

That's what people having public keys allows that goes a couple steps further than just the websites having public keys.

[0:12:10.8] JM: Okay. Makes sense. Public key encryption allows for these encrypted private messages or encrypted files. I can encrypt individual files within my Dropbox folder, so when I sync my local Dropbox with the remote Dropbox, it's going to sync encrypted versions of those files. Those messages, those files are getting encrypted using my public key. Also this public key infrastructure allows for signed messages, so that if you sign a message, you are using your private key and I can verify that it came from you if I have your public key.

In both of these used cases, I need your public key. In order to make these things work, you have to have a way of sharing public keys with people. If I get that public key from Twitter, or if I get it from an e-mail account I can't ever verify that first message which contains the key. This is the key distribution problem, and this is why people historically have had these key sharing parties where you show up in person and you give your public keys to people. How does Keybase approach the key distribution problem?

[0:13:30.5] MK: Yeah, that's a great question. I feel like we've seen a really exciting proliferation of security apps, like your telegrams and your signals. Those apps are super exciting, but they've all cut this corner and they basically just said, "We have this great protocols to make sure that people, once the key is exchanged properly can enjoy privacy." But as far as the key exchange, they are just saying, "Trust us. We'll get this right," and there's no way to audit that.

Keybase started with the other way of thinking about it. We thought we want this all, the key distribution problem first and then build apps on top of it. If Jeff, I want to look up your key on Keybase, what I would do is I would first find one of your known social media handles to look you up. I could look you up at anything you've chosen to improve.

Then once I've done so, my client who is running Keybase would do a couple things. It would first pull down the record from the Keybase's infrastructure about what the most up-to-date version of your profile is. That's a little bit tricky to get right, but what that is is it's a sequence of science statements saying – you'll start that statement with, "Hi, I am Jeff and this is my first device." Then you might say, "I'm adding a second device." Then you might say, "Revoking my first device and I've got a third device." Those are types of statements that you might see.

Also, "I am claiming ownership of this Twitter account, and I am claiming ownership of this Github account, etc." When I go to look up you, I pull all the information about your account, like this is chain of signatures. I play them back and make sure that each one is successful pointing to the previous one and that everything is signed by a device that should've been signing for you at the time.

Then after that's all done, I look at all these external proofs that you've posted and I make sure that they still exist in the places where you posted them. What that means is that my client is actually going to make a HTPS call to Twitter and to Github to make sure that the data that you said was there is actually there.

Once I put all that together, what I wind with is a set of public keys that identify you one for each of your devices, and also a knowledge that these other social media accounts that you say are yours are actually owned by you. You consented to associating them with your profile on Keybase.

This is a lot of steps to get right, but it's where we started and it's this notion of identity that we built everything else on top.

[SPONSOR MESSAGE]

[0:16:04.5] JM: At Software Engineering Daily, we need to keep out metrics reliable. If a bot had started listening to all of our episodes and we have nothing to stop it, our statistics would be corrupted. We would have no way to know whether a listen came from a bot or a real user. That's why we use Incapsula, to stop attackers and improve performance.

When a listener makes a request to play an episode of Software Engineering Daily, Incapsula checks that request before it reaches our servers and it filters the bot traffic preventing it from ever reaching us. Botnets and DDoS attacks are not just a threat to podcasts. They can impact your application too. Incapsula can protect API servers and micro-services from responding to unwanted requests.

To try Incapsula for yourself, go to Incapsula.com/2017podcasts and get a free enterprise trial of Incapsula. Incapsula's API gives you control over the security and performance of your application and that's true whether you have a complex micro-services architecture or a Wordpress site, like Software Engineering Daily.

Incapsula has a global network of over a 30 data centers that optimize routing and cashier content. The same network of data centers are filtering your content for attackers and they're operating as a CDN and they're speeding up your application by doing all of these for you and you can try it today for free by going to incapsula.com/2017podcasts and you can get that free enterprise trial of Incapsula. That's Incapsula.com/2017podcasts. Check it out. Thanks again, Incapsula.

[INTERVIEW CONTINUED]

[0:17:52.9] JM: If I understand correctly, this is a decentralized trust model where you have your initial key sharing that is backed up by the ensemble of your Twitter and Facebook and Github and whatever other social accounts and the bigger an ensemble you have, the more reliable your initial public key sharing is. Is that correct?

[0:18:19.5] MK: Yeah. I think that's correct. I think you want to have more devices, because you want to be robust to the loss of any one device. Unlike other services, when you lose your last device, your data is gone. Like there is nothing anyone in our company can do to recover that data for you, so you really want to avoid that situation. That's why we encourage people to add many devices to eliminate the chance that they'll lose all of them.

We obviously further encourage people to use endpoint encryption for instance on your iPhone or full disk encryption on your Mac, to make sure that even if someone does lose that device, if you lose your device and someone captures it, that they can't actually access the public key, that the private key and that device, because it's encrypted. That's why we ask people to use tad multiple devices.

It is decentralized as you say, in the sense that we don't want our users have to trust Keybase. We want our users to understand how the software works and for them to be okay with them, the software expressing the ideas that I'm expressing to you right now. But our servers start to lie to people, for instance if we become hacked or if some foreign government agency tells us that we need to lie about certain public dissonance or whatever, we will not be able to, because the software that's already in the wild is holding us to the earlier contract that we made.

In that sense, the trust is decentralized because if you're a bad guy and you want to take over someone's Keybase account, you'd have to both hack their Keybase account and also all their other social media accounts. In that sense, it's decentralized.

I can trouble with the real hardcore decentralization people, like the Bitcoin or Ethereum people if I say Keybase is decentralized, because we are not running our infrastructure on a blockchain. We run our infrastructure as a traditional web service. In every 12 hours, we post the most up-to-date version of our Keybase database to the blockchain. We use the blockchain to make sure that we haven't been coerced to show two different versions of what Keybase's database looks like to different people. But the infrastructure is really hosted on traditional cloud-based hosting services.

A lot of people would potentially argue with us about this decision. I'm happy to defend it, but I can't go on the record and say this is a fully decentralized application.

[0:20:45.1] JM: Once I have initialized my identity being synced with Keybase, or my multiple devices and my social media accounts all with my Keybase account, I can start to do some interesting things. The first interesting thing that you built was the Keybase file system. Explain what the Keybase file system does.

[0:21:08.1] MK: Sure. We thought about how people – if people did care about end-to-end security, where Dropbox or Google Drive can't know what's in your files, we thought about what the steps involved would be. The first thing you would have to do is look up someone else's public key, you'd have to encrypt a file for them using their public key, you had to put it into Dropbox, they would have to download it from Dropbox and decrypt it with their secret key and then open the file.

What KBFS is it's combining all steps of that chain into a simple operation, so that the users are basically ignorant to the fact that the cryptography is actually happening. It's just magically happening under the covers. So when we first started KBFS, it was more like one-on-one sharing opportunity. So if you knew someone else, maybe a couple people you wanted to share with, you can make a folder that the three of you could access, and then anything you put into that folder was signed for you, by you and encrypted for the other two people in the share to see, and they would have the reciprocal experience when they want to read the file out of that share.

We've recently improved that feature by adding the notion, very obviously not full notion of a team. A team, as far as Keybase is concerned is a group of let's say 1 to a 100, a 1,000 people that all go under some – that all can be captured under one name, like Poker Pals is one we use. It's like an informal team, or maybe a more formal one is Keybase_Corporation or whatever.

That's interesting now, because they could be a lot bigger. You don't have to list the people who with every file access as you did with our previous used case, and also that membership can change. But once you have a team of people, the same KBFS operation still work. You put something into the file, into the folder and it's now encrypted – signed by you and encrypted for everyone in the team, and then they get to read out if they're on the team and then verify that you wrote it.

That's basic KBFS both in the original case that we put out there and also the more recent iteration that has the team support for it. But it shows up on your Mac or Linux or Windows machine as a regular folder, but it's a folder with seemingly magic properties, because it's doing all this crypto under the hood without your applications or you really having to know about it.

[0:23:35.1] JM: A simple example is if I want to post a soup recipe to a folder and I want to share it with three of my friends, why don't you walk us through – just refresh for the people who were confused on the first explanation of creating a Keybase account. Let's say these three different people, they create their Keybase accounts, they create a shared folder, could you give us an overview of an abbreviated condensed version, because I know you already explained the process, but I think just to refresh people, give them a second chance to catch up for what they're going to be confused about.

[0:24:16.2] MK: Okay, great. So let's say a Mac and I'm sharing with two people, let's call them Alice and Bob. Software running on my computer when I first go to share a file, so I'm just as you said I'm just dropping a soup recipe into a folder. What my software would do is it will look up Alice and Bob on Keybase.

It will, as I mentioned earlier it will play back their identities. Alice has two devices, Bob has three devices, they have both the set of social proofs and my client will – if Alice and Bob are people I've never spoken to before, it will show me what their profiles look like. If they are people who I've spoken to before, it will just check to make sure that everything is still as it remembers. If so, it will take these Alice had two public keys, Bob had three public keys, let's say I have four public keys, so in total that's nine public keys.

My client will then cook up a shared secret. Let's say it will be a 32-byte secret and it will encrypt it for each of those nine public keys. Then it will use that shared secret to encrypt the file that the soup recipe that we put in the folder. Basically, we're not encrypting the soup recipe nine times, we're encrypting it once and we're giving the encryption key to everyone who should have access to that folder via public key technology.

The other thing my client will do before writing that soup recipe to the folder, is it will also sign it with the private key of the device that's doing the sharing. Then once that's there, Alice and Bob could go onto their devices, they first – basically, they do this whole thing in reverse.

Let's just say Bob finds the encryption of the secret key for his device, his device will decrypt it, that will yield the 32-byte shared secret key and then that will be used to decrypt the soup recipe. That's just one file. We have the same guarantees over a whole hierarchy of files, and so file names are encrypted, the context of directories are encrypted and signed. You add all of these components up and now you have this fully encrypted file system tree that Alice, Bob and I are sharing.

As far as the Keybase servers are concerned, we don't even know what is the soup recipe and what is the directory that contains the soup recipe. It's just all blocks that are totally encrypted and hidden from us.

[0:26:48.3] JM: That's great. Now that you've simplified that example a little bit more, I think we probably have motivated things for people who may have been a little bit lost at the beginning of the show. This happens in some of the more complex topics where people are listening, and in early stages of the show they're just like a little confused and then later on they're like, "I wish would've understood that complex topic that people were discussing at the beginning."

For those types of people, let's quickly run through just one more time how somebody – like what happens during the sign-up of Keybase, why this is an extra usability a modern – modernized for our modern multi-social network, multi-device world version of PGP?

[0:27:36.4] MK: Right. The things that are different here in Keybase is that in an example I just mentioned is that I have multiple devices, not one device and so does Alice and so does Bob. When I'm encrypting for – if I understand the question properly, when I'm encrypting for Alice, when I'm encrypting for Bob and they're encrypting for me, I have to make sure I'm encrypting to the right set of devices that represents them.

That device is always changing set of devices – just because over time we throw away computers, we buy new ones, etc. Your identity is just cryptographically, it's a set of personal devices that represent you and the set of public-facing social media names that you want to be made known.

The challenge here that Keybase is trying to solve without anyone knowing about the details is just making sure that when I go an encrypt for Bob, I'm getting the three devices that he's currently using, not the one he left in the cab last year. That this Bob has his three devices and that he has proven to me that he's the right Bob, not a Bob with Keybase is claiming he is, but the right Bob because my client is also checking that he owns the Github account and the Twitter account and the Reddit account that I know him by.

These are all part of the same computation that if any of these components are done improperly, it would be an attack vector. The attack vector might be that an evil Keybase – if our software didn't work properly would insert a bogus device for Bob, and that bogus device would be one that a government agency controlled. Therefore, the soup recipe would be known to the government agency because that fourth device that I encrypted for Bob mistakenly was actually not his device at all, but a bad guy's device. So our client protocol is making sure that that type of thing doesn't happen.

[0:29:31.2] JM: When I am signing up for Keybase, there is this like event where on all of these different accounts, on your Github account, on your Twitter account, on your Facebook account, you simultaneously publish what your public key is, right? There is a moment where all these things are synchronized, so when Keybase scrapes all these different sites and communicates with different devices, it can synchronize with all those at once and know that this is the ensemble of identities?

[0:30:01.9] MK: Yeah. The thing we ask you to post – Let's just go over, let's say the Github example. When you sign up for Keybase and you prove ownership of a Github account, we give you this little block of text that you then go and post on Github. What that text is, it's not just your public key. It wouldn't really make sense for just to be your public key, because then we'd be asking well, which public was it? It is the public key that you then lost, or is it like –

What we do instead is we ask you to publish the last signature you've ever made in your Keybase account. That basically captures the whole state of your account up until that time. What we asked you to post on Github will be – it will point back to your Twitter and prove that you did that previously. It will point back to the first device you ever used on Keybase, and maybe the revocations of devices you lost in cabs and provisionings of new devices you went out and bought, because you really want to get the iPhone 8.

The thing that we ask you to publish on Github is a cryptographic summary of all of that stuff. It's way more powerful than just posting your public key. When I go to look you up, I make sure that whatever you posted on Github is the same thing that Keybase is reporting back to me. So therefore, I know yet in a different way that Keybase isn't lying and making malicious stories up to compromise your identity.

[0:31:32.3] JM: That makes sense to me. For people who are still confused, I think you can go back to the previous episode that we did, or you can look on the Keybase documentation, which is really good; has some great explanations of what's going on here. Let's get back to the file system. How are these files stored and sync to give me the client-server model?

[0:31:55.5] MK: Sure thing. The easiest way to describe it is that there's basically – this is not a Dropbox model where basically – so the Dropbox model more or less is whatever you put on your machine, Dropbox lazily synchronizes to its servers. After about a 10-minute – however much data you have after that window, all the data is synced in probably arbitrary order between your computer and Dropbox.

Our file system has a bunch of different properties. I think the first property, which is pretty compelling is that you can have 10 terabytes of files on Keybase and only 10 gigabytes of storage on your device. That's actually quite useful for people who have more and more – it seems like our SSDs are not growing – they're not keeping up pace with the amount of data we have.

This is one thing that's very nice about Keybase, where you don't have to have as much local storage as you do files. But the way the Keybase system works is that our file system client is actually at the operating system level, it's understanding every system call that your application is making.

The first system call your application might make, let's say, or word processor would be to open up a file. That means the Keybase client running your machine gets this open call, and then sends that off to the server – sorry, the open – let me get this right. It's a little bit tricky. Probably means that the directory that the file was in is modified, so then the Keybase client sends that updated directory, encrypts to the server as this encrypted block, and now the directory with that new empty file in it is now on the server so that other people who want to see it can download that new version of the directory and see that file exists. That's the first system call.

The next system call might be like a write. Let's say your word processor, you type a bunch of words, you hit save. That probably means there's going to be a write to the file and probably like a sync of that file. What that means is a new file has now come into life. That's going to be encrypted, there's going to be a block, probably one of the small, there's going to be one encrypted block. That's then sent to the Keybase servers. The directory probably has to be updated, because the file in that directory is a different file, so that probably the directory itself has to be encrypted and sent back to Keybase and then so far and so forth – so and so forth, all the way up to the root of the directory tree.

Basically, all that happens kind of piecemeal until eventually this file is put down onto a storage and the Keybase servers have it and other users can now view the file. It's called a Merkle tree basically and the file system itself is represented as a Merkle tree, which is quite useful for performing cryptographic operations over the whole tree.

When you actually sign the state of the file system, you're not signing each individual file, you're signing the state of the whole tree. So that if someone wanted to tamper with your files and let's say change a filename or change what file a filename pointed to, that would be detected as a result of signing just the root of the tree.

[0:35:12.7] JM: When you talk about that Merkle tree that defines the file system, are you talking about a user's file system, or is this a global Keybase file system?

[0:35:21.0] MK: This is the global Keybase file system. Users are able to write into certain sub-trees of it, and so that suppress a few directory I had with Alice and Bob, that's a sub-tree of that file system. Above that, all those sub-trees for each group of users is combined to get one global Keybase file system, which is the whole thing is basically captured in one 32 byte hash. That's like the state of the whole Keybase global file system.

Of course, access control is very important here. So just because it's a global file system, it doesn't mean everyone connects as all parts of it. The crypto ensures that you get to see only the right parts of these global files.

[SPONSOR MESSAGE]

[0:36:10.6] JM: You are programming a new service for your users, or you are hacking on a side project. Whatever you're building, you need to send e-mail. For sending e-mail, developers use SendGrid. Send Grid is the API for e-mail trusted by developers.

Send transactional e-mails through the SendGrid API. Build marketing campaigns with a beautiful interface for crafting the perfect e-mail. SendGrid is trusted by Uber, Airbnb and Spotify. But anyone can start for free and send 40,000 e-mails in their first month. After the first month, you can send 100 e-mails per day for free.

Just go to sendgrid.com/sedaily to get started. Your e-mail is important. Make sure it gets delivered properly with SendGrid, a leading e-mail platform. Get started with 40,000 e-mails your first month at sendgrid.com/sedaily. That's sendgrid.com/sedaily.

[INTERVIEW CONTINUED]

[0:37:27.0] JM: What are those parts? If I'm Jeff and I've got a shared recipes folder with Alice and Bob that originated in Bob's home directory, what is my local snapshot of the file system look like? How am I interacting with it?

[0:37:47.2] MK: Yeah. I mean, when we first implemented it you locally didn't store any files. All the blocks that used to make that sub-directory were loaded from the network as you needed them and written back as you wrote them and nothing hit your local disk at all. That was the one of Keybase.

Since then, we made a bunch of different changes to improve performance. One change that we made is that we have what's called a file system journal, so that if you're doing a whole of operations, like let's say you're uncompressing a compressed zip file, that's a classic example where like one click of the mouse can cause a 100,000 small file system operation.

Now our most recently client will just write them to local disk, buffer them up and send them all in batches, so that each of those 200,000 operations is in a roundtrip to the server. That's one thing we've changed recently for performance. The other thing is that if we see you're accessing the same files over and over again, we'll just prefetch on your local client those blocks that correspond to the files that you access the most.

These are just optional improvements that are not there for correctness, but just there for performance. Obviously, we're still trying to find the right tradeoff between performance on the one hand and not hogging too much storage on the other. There are the bunch of sliders you could slide in between there, but that's more or less our model.

It's a totally different model from the Dropbox model. It's probably closest to the Dropbox infinite model, which they've released last year for their business customers, which is a little bit more on demand. You don't have to selectively sync ahead of time this whole part of your shared files, you just go and access the files and they magically appear. That's like the easiest comparison I can draw to what's currently out there in the market.

[0:39:40.7] JM: Sorry, but you are storing the files on Keybase's servers?

[0:39:45.8] MK: Yes. We're storing everything and we use Amazon web services. Everything is stored as encrypted signed blocks. Blocks can be anywhere between a 100, let's say a 1,000 bytes for small files or small directories, up to like, I think the maximum block size is 8 gigabytes – 8 megabytes, or 8 gigabytes? I think it might 8 gigabytes. From there, we can make much bigger files by putting a bunch of these blocks together and making bigger files.

Actually I'm wrong. The maximum block size is 512k. That means if you have let's say a video that – I'm not going to do the math right, but let's say it's a terabyte long, that means you'd split this file up into 512k blocks and store each one of those individually on the Keybase servers. The person you're sharing with would download all those blocks and stitch them back together and present a file to your application, so the application can watch the video.

[0:40:46.8] JM: Okay. On my local device I've got – is it basically like a directory structure? I don't have the actual files, but I've got like a directory structure. Every time I open a file that's on that directory structure that I have access to, like the recipe, it's going to check and make sure that my local – sorry, I guess the private key I have locally can be used in concert with one of the public keys that I signed up for this file with, because in that example you gave with Alice and Bob where they've got nine keys between them and maybe I've got three keys myself and that's 12 keys. So there is 12 different devices that can access that file, and as long as I've got one of those keys that gives me access to that file I can retrieve it from the server?

[0:41:37.7] MK: yeah, exactly. You can retrieve it from the server, A. B, you can decrypt it. The server doesn't know if you can decrypt it, it just hopes you can because it certainly can't. Yes, and then once the blocks are decrypted, indicate to the software running in your machine how to stitch the blocks back together to make a file system to make a directory tree, to make it look to your local operating system, like you're dealing with a local file system.

That's an illusion. It's just something that's just presented at the very last minute to the applications to make it look like regular files. But really these are encrypted synced signed files without the applications having to know it.

[0:42:16.6] JM: What if I try to access Sally's recipe for a green bean casserole and I don't have permission to that, but I'm accessing it in a place where it actually exists on the Keybase's global remote file system, am I going to get some blob of blocks and then I'm going to fail to be able to encrypt it?

[0:42:38.6] MK: Yeah. Actually we're doing a little bit of belt and suspenders here. We're not even letting you download that block to begin with. You'd have to prove first to our servers that you could access – that you were one of the people listed in the access list for that file and you'd do that through just a standard authentication mechanism. Then once you did, you'd be able to download the block and then decrypt it.

In the case you mentioned, you'd fail even to download the encrypted block to begin with, because why should we let you even do that? It seems like a safe thing to do. If someone broke into your servers and published all of the contents of our three buckets to the world, then we'd obviously lose that. First we'd lose the suspenders, but we'd still have the belt, which is that obviously you need a private key to decrypt, to actually look at Sally's recipe.

[0:43:28.9] JM: What goes on in that communication between the client and the server there where I've got to be able to authenticate somehow to be able to even prove that I can get the right blob of stuff?

[0:43:41.7] MK: Well this, we have a very committed answer for. Authentication is done with your signing keys. The server presents you a challenge and say, "Hey, Jeff. Please sign a statement for me." Then you use your whatever secret you have to just sign that statement and send it back. The server knows what are the public keys that are to represent you and therefore, it can say, "Yup, that's Jeff. He's allowed to access this data."

It's not too different from authentication you would do with your bank, but it's better because it's using public key. It's probably most similar to like SSH authentication where if you SSH to another machine, you're probably not typing your password, you're just using this automated experience where you're using your local secret key to sign a challenge, to prove you are who you say you are to the server.

[0:44:32.9] JM: When you build a new file system these days, how much can you take off the shelf?

[0:44:38.1] MK: It's a good question. We actually wind up basically not taking much off the shelf. We take our crypto-components off the shelf for sure. So this is a decision we made early on, that's just been so far and away the right decision.

We used the salt of NaCl library run by Dan Bernstein to do all of our crypto. It's just every week you'll see another vulnerability in other libraries, like GPG has all of these timing attacks that people come out with. Today, I believe there is a problem with the Harvard-generated PGP keys and some weird implementations of PGP devices.

We've just have not even close to a problem with this library and we've been just so grateful for it. I mean, he's made so many correct decisions and all these vulnerabilities keep reinforcing that in our minds that this is just great engineering.

That we took off the shelf, we are grateful for Go and for the incredible libraries that they have, because I come from a background where we used to write everything in C++, and the issue I always had with C++ is it was so hard to assemble a group of libraries that worked well together, because everyone had a slightly different model of threading or a different memory protection and garbage collection model.

The nice thing about Go is everything in the ecosystem works together so seamlessly. All that stuff we've taken directly off the shelf. When it comes to a cryptographic file system that's distributed and that works with our identity model, we really couldn't use anything aside from those components I just spoke about. So we had to write a lot of it ourselves.

[0:46:24.8] JM: Tell me something that was really hard about building the distributed cryptographic file system that is Keybase file system.

[0:46:33.6] MK: Yeah. I mean, it's an ongoing challenge. One of the challenges of a file system that's accessed in this way, where it's like this on-the-fly access is that the operating system is giving you pretty stringent requirements. On OSX for instance, we have to answer system call in 19 seconds. If we don't, there will be a message sent back to the client that says like, "Oh, your file system is not available."

Obviously, if you're plugged into a great connection that should not be a problem, but if you're on a laptop and your Wi-Fi cuts out, it's a little bit more of a problem, because you could easily have 30 seconds of a disconnectivity that is just normal, but we have to find a way to mask that as far as the user is concerned.

That's maybe one of our biggest challenge is just dealing with that 19-second timeout. I believe Linux has like a 60-second timeout, so that gives a little bit more time. We'd have to come up with a whole bunch of strategies under the covers to, if there is a network condition that's not great to mask that and let the application keep working. That's definitely been one of the biggest challenges.

[0:47:40.7] JM: You touched on this earlier that you integrate with the blockchain, the Bitcoin blockchain to ensure integrity of the Keybase file system over time.

[0:47:49.5] MK: Yes.

[0:47:50.5] JM: Come back to that.

[0:47:51.2] MK: Sure. Everything in Keybase, from the file system to the depictions of people's identities, everything is stored in a series of what are called append Only Data Sructures. What that means is that we wanted to avoid the case in which an evil Keybase could drop things off the end of people's identity chain.

Here's a good example, I leave my phone in the cab, the cab zooms away, I go to my computer and I say, "Please revoke my phone." Keybase just doesn't – Keybase gets that request, but just doesn't propagate it to the rest of the world. It just hides that and pretends I never said it.

Now that person who gets my phone who finds it in the cab can then access all of the Keybase services as if they were me, even though I tried to revoke the device. That's something that needs a good solution. The solution more or less is what we call this Append Only Data Structures, to make sure that an evil Keybase is never in a position where it's able to roll back the state of the world to a previous state, that once it publishes a state, it can only make changes and modifications to that state. It can't pretend that something that happened that it said happened didn't happen.

We built everything in Keybase with an Append Only Data Structure. That still leaves one attack open. One attack is that if I'm showing a version of Keybase to you and I'm showing a different version of Keybase to your friend, I could get away with that as long as I never try to merge the two versions together.

It's like basically, I could fork the state of Keybase. It could be appending onto fork A and appending onto fork B, and this is like a super evil Keybase. It could just make sure that both sides of the fork never know about each other, and then it can get away with that.

There's one additional precaution we take, which is that Keybase has to say, "This is the right version of Keybase in the Bitcoin blockchain," and there is only one of those. Therefore, that kind of fends off this last attack, this forking attack because Keybase will be forced to put either fork A or fork B into the Bitcoin blockchain. It can't do both. Therefore, anyone who has access to the Bitcoin blockchain can know they're on the right fork.

Everyone talks about Bitcoin for solving a million different problems. This is in our view the most important problem it can solve, and everything else we can do off the Bitcoin blockchain and get all the security guarantees that Bitcoin allows.

[0:50:26.6] JM: An elegant solution. I know we're up against time. You have had some pretty cool product releases recently with Keybase Teams and Keybase Git. So Keybase Teams is described as like slack for the whole world, or a team Dropbox where the server can't leak your files or be hacked. Keybase Git is Git on top of Keybase. Could you quickly talk through these products and how they – I guess, also how they represent the big vision of Keybase.

[0:51:01.5] MK: Yes. Our big vision is that we want people to use Keybase for applications today. I think our biggest, biggest vision for Keybase is we want to build those graph of cryptographic identities where you know who my friends are and these are not just Facebook saying it's this link, but this is the real cryptographic link that nobody could've possibly faked, and we want people to know what all of my devices are.

That's the ultimate overarching goal with Keybase. But before we get there, we really want people to use this product today for problems that they have. I think Teams is something that took us forever to build, because it was actually quite complicated. We got to Teams, because the original version of Keybase wasn't sufficient for a lot of people's needs.

It's easy to say like, Bob, Alice and Jeff are the small group of people and they're never going to change, they're never going to add Sally. That's what our original version of Keybase is good for, but that doesn't work at all for people's companies where obviously employees come and go. It's nice to have a name for something. That's where the idea of team – a team comes in.

Now we talked several times about how a person's identity is this sliding window of device changes. It's only after you play them all back in the right order that you get to the right version of what the person is at the present. A team is the same thing, just one layer up. A team is a combination of people, where you and I start a team and we add Sally, then Sally adds Bob, then I decide to delete Bob and so on.

There is this very parallel structure in a team where the composition of the team is changing over time, but it's always cryptographically verifiable that is the right team of people. You can imagine a server attack here where, let's say you and I are on the board of directors of a public company and we're discussing some sort of upcoming big decision, and our nemesis is trying to get access to this data.

One thing they could do if they co-opted Keybases to insert a fake person into the team that just gets to – a fake director who is just getting access to all these information. Unless your system has some protection against that attack, all the encryption the world doesn't help you. What Keybase Teams does is it allows the users of the members of a team to audit and verify that the correct team members are in the team at the given time.

That's a snapshot of why it was so challenging to build, but once we build teams we just unleashed it on all of our existing product. I mentioned before, it works with KBFS, it works with our chat product. That integration was actually pretty easy. It was just the building out of the team's infrastructure and that was the challenging part.

That's something that we think is quite transformational. Anyone who is using something like Slack inside their organization – I mean, if it were me I'd feel a little bit worried that if Slack has ever compromised, I'd lose a lot of very important data. We want Keybase Teams plus our chat feature to be like a good Drop and Slack replacement, with the important added distinction that everything is cryptographically verified and end-to-end encrypted, which is we think quite a unique feature.

The other thing that you mentioned Git, Git is kind of a different interface into a file system. There's a couple more steps, if you were to write a soup recipe and sync it to your friends on Git, you'd have to type Git commit and then you have to type Git push, or you'd maybe use some sort of graphical user interface to do that.

It's a couple more steps. However, it's what people are probably doing now anyway when they start to collaborate on files, especially source code that has a lot of different authors where you also want to have a very clear way to review history, and to see when a bug was introduce or to resolve conflicts. We think that – I mean, I don't have to sell GIT to your audience. I'm sure everyone loves it. But we just wanted to have a very easy way to use Keybase file system for Git.

That's all that Git was, it was actually a pretty future for us to build, because so much of the foundation was already laid in the previous features that we talked about. Of course, it works with Teams and it works with personal repositories too.

I mean, I for one I'm a tremendous user of this feature. We run a web service, we have a lot of API keys, we have a lot of configuration data. We do not want to push that data to Github, because the Github is great for things like code reviews and file browsing. It is not a place I'd ever feel comfortable storing my secrets. We use Keybase almost exclusively for anything that has secret data in it, so that's either chats or configuration files, I don't know, kind of internal management documents for managing our finances or employee offers and stuff like that. Those are things that we feel much better if we keep it secret and manage on Git on top of KBFS.

Yeah, those are our two most recent features and we're super excited about them, because we feel like they're good dropping replacements for what people are doing now and they have these amazing security properties that are quite challenging to provide.

[0:56:17.2] JM: Max Krohn, thanks for much for coming on Software Engineering Daily. It's a really ambitious project that you're working on with Keybase and I love reporting on it.

[0:56:25.8] MK: Thank you, Jeff. Thanks for the thoughtful questions and for giving me time to talk about this product. Thanks so much.

[0:56:32.3] JM: Absolutely.

[END OF INTERVIEW]

[0:56:37.6] JM: Thanks to Symphono for sponsoring Software Engineering Daily. Symphono is a custom engineering shop where senior engineers tackle big tech challenges while learning from each other. Check it out at symphono.com/sedaily. That's S-Y-M-P-H-O-N-O.com/sedaily.

Thanks to Symphono for being a sponsor of Software Engineering Daily for almost a year now. Your continued support allows us to deliver content to the listeners on a regular basis.

[END]