Eric: Hi, Professor Falk was the fifth episode of The Encrypted Economy. Now I encourage you to listen to that episode maybe before listening to this one, and particularly if you need an introduction to privacy enhancing technologies. Yep. Professor Falk has a particular gift for explaining complicated concepts in an understandable manner.

And since our fifth episode with him we covered a lot of ground with regards to privacy enhancing technologies. I think I've covered that elsewhere. Um, Now I've kept up with his work since he came out to the podcast and particularly with regards to his most recent article on oblivious Ram, which really caught my attention.

I had to invite him back at the heart of this is data leakage and inferring data from access patterns. Even when it's encrypted this not only protects the privacy of the subjects and data calls, but provides less clear. For advanced persistent threats, ABTS from malicious actors, considering the range of big data applications for multiparty computations.

These advances in technologies come not only with computational limitations, but also privacy and security limitations as well. Now, more interesting is how obliviousness can vastly improve these risks particularly for highly sensitive or confidential information across competitors or across regulated datasets framed differently.

What does this make possible? This type of technology. So a further any episode, we break down the differences between circuit base versus Ram based encryption, the use cases for each as well as balancing efficiencies and security in multi-party computational architecture. So we also touch upon efficiency considerations for implementing encryption and secure multi-party computation.

So if you have at least a basic understanding of secure multi-party computation and any interest in learning more, you have to listen to this episode. And I encourage you to also read professor Fox paper on oblivious Ram linked into the show notes and to share it towards the end. We shift towards another area which Brett has been focusing on, which is credential management and how to live.

Uh, verifications and credential management with a blockchain. Uh, another interesting episode with FOC, so excited to bring it to you. If you enjoy it again, please share in the interim, I bring you professor Brett Falk. Welcome to the encrypted economy, a weekly podcast, featuring discussions, exploring the business laws, regulation, security, and technologies relating to digital assets and data.

I am Eric Hess founder of Hess Legal Counsel. I've spent decades representing regulated exchanges, broker dealers, investment advisors, and all matter of FinTech companies for all things touching electronic trading with a focus on new and developing technologies.

We are happy to have returned to The Encrypted Economy podcast professor Brett Hemenway Falk. So Brett, welcome.

Brett: Thanks. I'm glad to be back.

Eric: Yeah. So for, for the listeners where we're going to skip a couple of things we normally do because Brett's already been on and in fact, he was sort of the building block for much of the. Subsequent episodes that we did in privacy enhancing technologies.

He was the first. So he was, you he was a pioneer in this space for us here. Um so if you haven't listened to that episode or you forget either way, I think that's one of the episodes definitely worth a repeat get to get his background, get his origin story. You know, if you want to brush up on some of the basics of multi-party computation, homomorphic, encryption, privacy enhancing technology.

Go there. Um, but we're just going to kick right into it. Um, so this was great. I was so excited to have that bread agreed to come back on. Um, and of course I'm reading, I'm looking at his academic papers over the last year. He hasn't stopped. He's a, he's a force of nature. He's you know, intellectually curious about everything and, and deep dive and very intelligent.

What a boon to the overall space. We had a long talk before even doing this podcast where okay, we gotta find a way to, to cut this down for the podcast. But with that said we're going to launch right into it. And w we're going to talk about privacy enhancing technologies, but we're going to start off with you know, sort of a question of the state of privacy enhancing technologies specifically as it relates to cost improvements and computational limitations.

Brett: Yeah. Um, yeah, that's been a barrier for a lot of these. These techniques is that they are very slow, right? And so we can talk about a couple of different things. Um, multi-party computation, MPC like fully homomorphic encryption and then hardware based solutions, things that use like trusted execution environments, like SGX.

So right. Fu is MTC. You're trying to compute on some private data and you have a group of people, and now they're going to run this computation. They all have their own private data and they all have to communicate with each other. And these protocols first is they grow with the number of participants, they grow quadratically. So if I have five people, they all have to talk to each other. And so if I have 10 people now 10 people all talking to each other as a lot more than five people, all talking to each other. So with the multi-party computation, they work much better with smaller groups of participant.

Whereas something like fully homomorphic encryption in principle, you can encrypt your data. You don't need a whole group of people. If I have it works well, if there's just maybe one person who wants to outsource their data to the cloud and have the cloud compute on it. But both of these systems, they're software based systems and they have some huge overhead.

And rule of thumb is maybe it's like on the order of a thousand times slower or something to use one of these secure multi-party computations than to do this same computation in the clear. And again, it depends a lot on exactly the type of computation you're going to do. Um, but this is something that is a big problem and it's a real barrier to adoption for these things because.

It's just very slow. And so you can do a lot of these much smaller computations things that would have been very fast, right? If you were going to try to do some computation on private data and it was going to take you a millisecond to do it in the clear, well now a thousand times slower, it only takes you one second.

And if you're only doing this occasionally this is no problem at all. But if you're doing some kind of big data type of computation, which was already sort of pushing your computational limits you basically have no hope of doing something like that. Using current NPC are fully homomorphic encryption solutions.

Um, and the hope is that all these things will get faster but that is a big barrier now. Um, and one of, one of the issues I think with this is that the. Uh, computations for a secure multi-party computation, they have to be oblivious. And I think there's something we can talk more about, but basically the idea is that you have to hide not only the data, but also the control flow, that execution of what's happening.

Um, because this can't depend on the data. And so for example, like a really simple example, I think of showing why these computations are much slower is imagine you wanted to do some search over private data where, you know you have an encrypted array and I want to find the first element in your array, where you have an encrypted array.

People with salaries or something with their salaries. And I want to find all of the indices where people salary is more than a hundred thousand dollars a year. If you do some kind of computation, which is doesn't decrypt, right? So all the data remains encrypted, but you're doing some kind of secure comparison, right?

I give you some encrypted query. I give you an encrypted query that says, I want to retrieve values from this encrypted database. Um, and I want my encrypted query is some salary range. I want people who salaries above a hundred thousand dollars. If you start going through this database, even if the values are encrypted in the queries are encrypted, you at least know what records you're touching. And if you go through this database and at some point some parts of the compensation, you spit out some encrypted values and some parts you don't, then you know that something different was happening, if at the end of this comp, or if you go halfway through this database and then you stop. Um, and don't do any more, then you learn again something about my query, right? So if as a simple example for imagine your database is fully encrypted, but it's sorted and you happen to know it sorted.

And I give you some range query of. All the salaries of all the names of people who salaries over a hundred thousand dollars, right? If you go along your database, you're going along your database, doing a secure computation. Um, and at some point you hit this threshold, right? Cause your database is sorted and now something different starts to happen at this threshold.

Or if you cut off the computation, if you sort your database from highest to lowest, and if you in a normal search you'd sort start. You'd say I'm not processing any more. Once I get below this threshold, I'm not processing any more because I found what I needed. I know there's nobody else in my database.

That's going to do this, but in a secure computation, you can't stop like that because that would leak some information. So even if all of the records are encrypted, the fact that your computation sort of stopped early would leak information. And so that's not allowed in these models of secure computation for that underlie secure multi-party computation.

And so what that means is for these even simple algorithms, like searching or sorting or something like this, right? Um, you you always have to have the worst case runtime. So in like a sorting algorithm, if I give you data, that's already sorted, but it's encrypted. You have to go through the full sort.

You can't take advantage of the fact that it was already sorted. Um, because. If it was like partially sorted and you stopped early, that would leak some information. Um, and you could think of it you know, I think a good example is like playing cards, right? So were playing poker or something.

And if I happen to know, you really liked to sort your cards based on suit and you know and then by rank, after that you draw some cards. And if I know you're going to sort cards like this, even though your cards are sort of encrypted to the, I only see the back of them. If I see you moving cards around a lot, versus CU not moving cards around I get some information about the distribution of your hands. Even though I never saw anything to the cards, but based on the movements you made, and this is like a good analogy that even when data's encrypted, you can often see what the data movements are like, where this encrypted data is still stored on your disc and it has to move around.

And if these movements depend on what the underlying data is, then these movements themselves leak information and the encryption itself doesn't help you there. And so you need to make sure that not only is the data. But the algorithms are what are called data oblivious that they always do the same thing, no matter what, like the normal sorting algorithms that you think of are not data oblivious.

If I gave you a deck of cards and asked you to sort it, you start looking and moving around, but you wouldn't move things that don't need to be moved. And so that wouldn't be data

oblivious. And so if you want to do a secure sort, not only does everything have to be encrypted, but you have to do a data oblivious sort, which means that you're going to do the same sorting operations, no matter what the order, the the data came in and this kind of data oblivious sort is inherently going to be sort of worse than a regular store, because it can never stop early.

Um, and so you always get the worst case runtime. And so this is one of the barriers to this kind of secure computation. Um, and it, yes.

Eric: No. It's interesting. If you think of a D to use a different example, I don't know, w with a different result, though like you go on Google search, Google Chrome, you do enough searches over time.

They're going to hone to your particular searches. And so that whole mechanism of being more efficient is already sort of built into your searches. I run a search, like I'll ask my kids or somebody else to run a search. And I'm like, how'd you get that? I got exactly what I needed, you know?

Google's got an algorithm that they're honing into you and in all that efficiency is lost and the impact of that efficient of that inefficiency sort of obviously limits its use cases and also increases its

Brett: computational overhead. Yeah, exactly. And so again this kind of secure searching is right as something that you would like to do in a lot of situations.

Um, and you'd like to be able to do it sort of faster and what I was sort of mentioning in this data, oblivious world, if I just want to do one search, I'm searching if there's some encrypted database and I want to do a secure search on it, I have to touch every record in order to maintain privacy, because if the search ever stops early and doesn't touch this record, then I happen to know right.

That, that wasn't the right. That that it leaks some information and maybe a tiny amount of information, but it does leak some information about the encryption on

Eric: this notion of the cost and computational limitations. Um as we move toward the you the quantum computing that, that goal or just higher processing speeds, does that make the problem.

Uh, obviously that mitigates the problem. Is there a pathway where that becomes w we're secure multi-party computation and even homomorphic encryption? Homomorphic encryption used to be what, like a million or a billion times. It was theoretical. It existed in theory, before it ever existed in practice, do we ultimately get to that point where it becomes significantly more efficient

or

Brett: so hardware helps everything, this hardware gets better, right? Everything gets back. But the sort of relative ratio between secure computation and insecure computation doesn't necessarily decrease with hardware, right? So as you get more hardware you just store more photos on Instagram and searching them. Insecurely is still very hard.

Um, and so then searching securely is even that much harder. Um, the like absolute scale of the computations you can do just definitely increases as the processing speed increases. And for secure multi-party computation, it really matters about the network connectivity too, because if you have a lot of different people talking to each other and they're sending messages back and forth it's just much faster if the bandwidth is not limited and the latency is lower.

So this absolutely helps, but it doesn't sort of get you closer to being able to do everything. Securely if that makes sense.

Eric: But, but do you see, like how many years before have you, have you seen any studies or do you offer, do you want to offer any guesstimates about the the next five to 10 years

Brett: for your multi-party company?

I think right now we're already getting to the point where you can do you know, moderate sized calculations in a reasonable amount of time. Um, and you know, if you have something where like a lot of the. Projects that I've worked on in the past rate have looked at, you have different say companies that have datasets and they want to do some joint analysis on this data set.

So you have customer records or something. So I think a nice example is like insurance companies. So you have an you know, insurance companies that have records about. Their individual patients. Um, but patients move insurers all the time. And nobody keeps a single insurance company for a really long time.

And if you want to ask a statistical question does the preventative care we gave you when you were 18? Does that reduce the amount we have to pay for your care when you're 60? That's a really hard question to answer because no, no insurer has the continuous record of somebody when they're 18 and when they're 60.

And so you'd like to link across these data sets. And if you want to do something simple, like a linear aggression or something, or just even like a cross correlation of let's look at these two categories, people who had preventative care when they were young and how much do we pay for them versus people who didn't have preventative care versus how much we pay for them.

You'd like to aggregate data from lots of different data sources. Um, and you'd like to run this kind of calculation, but yeah. Great. These datasets can be big, right? They can have millions of records or something, but they're not sort of big data by what we call big data now. They're not. You know database with a hundred thousand records or a million records is not sort of pushing the limits of standard computation now. And so this is something that we can actually do with secure multi-party computation. And so this is something that's really nice. And especially because this is something that doesn't have to be done in real time, if it takes you five minutes to do this computation, or if it takes you an hour to do this computation or at the companies, or they're okay with that, right? They're not, it's not like a user interface that where the answer needs to be there. And if you're a millisecond slower, your user's going to go away, right? Like you'd have some research analysts who wants to do this and they start this job running overnight and tomorrow morning they get the answer and it's pretty cool because it was an answer that they just couldn't have gotten before.

Right, which is very different from like the Google kind of experience where you say, I would like to crawl the web of I don't even know trillions of web pages, how many web pages there are and give you some response. And if I'm five milliseconds slower you're going to go into my competitor.

That's, we're, we're very far away from that. But we're actually in a pretty good realm for analyzing databases with hundreds of thousands of records or a million records, especially if they don't need to be done in real. Um, so yeah, no,

Eric: sorry. I didn't mean to cut you off. Um, but I was going to actually note your a perfect example of this.

I think you wrote a paper last year privacy, preserving network analytics you know, specifically for a financial network models. I think the anticipation was used in traditional finance regarding risk management solutions and sort of getting a sense of risk across you know, multiple financial institutions without leaking their own particular holdings.

Um presumably yeah, that wouldn't be a real-time model, which I guess is also good.

Brett: Um, yeah, exactly. So this paper, we were looking at these traditional financial models where you have some number of you know, financial institutions like banks, and they have cross holdings in the form of debt and they have crossed holdings in the form of.

And you want to ask questions often about financial stability. So that's a lot of questions about stress testing. So you say what would happen if you know, everybody's assets dropped by 10% would some people default on their debts and if they defaulted on their

debts, then you have to worry about there's sort of contagion, some kind of collapse of I like a cascade of collapses where I default on my debts.

Then the next bank that I owe money to might default on their debts. And you could have a whole chain of failures and this kind of this type of analysis requires you to have a global view of the entire network because nobody can do that locally. I can say what would happen to me with debts? I would default on if.

Assets dropped by 5%. But what if all of the people who owed me money, what if they stopped paying me right? Then I might default on even more. And I don't know whether they're going to default or not. And so you, the only way you can figure this out is by looking at the network as a whole, but in practice, a lot of the individual institutions certainly don't have a global view of the network, they know what their own assets and decks are, but they don't know what the full balance sheet of all their competitors are and all their counter. And I'm actually, even now with a lot of the regulatory stress tests that are being done, they're done locally. And so they don't actually take into account these cascades of failures.

They ask each bank individually, Hey, what would happen? Would you default or not if you know under these different scenarios, but they don't actually look at the whole network and to look at the whole network, you have to aggregate all of this private information. And so this is actually a really nice use case for secure multi-party computation, because if you only care about the large financial institutions, which may be reasonable if you're worried about these sort of macro of.

It's there aren't that many of them, maybe you have a hundred banks or even if you had a thousand banks, right? This is something that's sort of within the realm of feasible today. And you want to ask these questions. You want to ask these kinds of counterfactual questions. Hey, what if everybody's balance sheet dropped by 5%?

Would we see a cascade of failures or not? And that's like the kind of question like to answer. And now each individual institution has its private information, which is all of its debts. And and that's what they feed in. And you do this kind of big calculation and outcomes, some answer, and you can tune it to release whatever you want.

You could just say something general generic about if under this scenario, there were this many failures and not identify people. And that gives them a little more privacy. Or you could say if it's important to know who the people who failed were right, you could have the secure computation.

Or you could have the secure computation only notify the individuals themselves if they were going to fail and not release that data to anybody else. And that would be sort of,

again, maybe more privacy preserving, because I think as we talked about last time, this notion of privacy is also very subtle in terms of what does the output of the computation.

And the cryptographers view of secure multi-party computation is that the, whatever the output of the computation revealed was okay, like that's the cryptography can't hide the output. You wanted to get the output. So if the output includes a list of everybody who failed under this hypothetical drop in asset prices the MPC is not trying to hide that.

It's trying to hide and it's guaranteed guarantees. You learn nothing more than. But even that might be embarrassing to you. If you're a company you don't want it publicized that if asset prices dropped by 5%, you'd fail on your debt. So you might want to say the output reveals even less. Um, but that's a question for the designer of the computation.

That's sort of outside of the cryptography, doesn't try to hide that through cryptography, just hides the underlying inputs to the computation and it says, whatever will make the output whenever you want. And the output could be a list of everybody who's fails, or it could be just a count of the number of people who failed, which would reveal less information, or it could be different outputs to all different people to say to individuals had you failed, or you didn't and not tell anybody.

And that might be less useful to society, but it would be more privacy preserving. So this is a question sort of like outside the secure computation, but we can build a secure computation to do any of these things while keeping the underlying balance sheets of all the company's private. And you can do that.

And now, again, this is a computation that probably would take on the order, we did some benchmarks for about you know, a network of 20 banks or something. And on the academic hardware that we have, which is not all that fancy, it's reasonable on the order of hours to do this.

Um, and if you were willing to pay $50,000 for some hardware you could probably increase this by a decent amount. And if you wait, two years, $50,000 will buy you even more hardware and it gets even more reasonable. And the number of banks that you actually care about, doesn't really grow maybe that much in the world.

It's not you know, the number of big banks doubles every year, right? You still just have the top thousand banks control everything. Um, and so this is a kind of computation that's actually really reasonable to run. And if it took you overnight to run this calculation, it's not a big deal.

Like you don't need to know that. You know, in milliseconds or even in minutes. Um, so this is, I think a really nice application of this type of secure computation. And again, it doesn't have to be we chose a very concrete example about looking at you know financial

contagion in interbank networks, but you can ask these kinds of questions for almost any type of you know, group or consortium of companies.

Eric: Well, in, in, in the context of a regulated institution that does have to do a risk analytics on systemic risk and report it. Um th this would better enable maybe the smaller participants to adjust their models within cycles of examination, or what have you, or not have to wait for published reports, but to actually, maybe real-time manage it a little bit better and better situate themselves to sign that could change quickly.

Now, if you're like a large bank, like JP Morgan, Your sheer size probably gives you a pretty accurate barometer of the entire marketplace. And that information would be obviously very valuable to a lot of participants in the network, but JP Morgan is going to be like, okay, here's my balance sheet.

Go figure it out. Um, but you it reminds me years ago when I was general counsel of the stock exchange, there were a couple of very large clients that were uh, if somebody wants to explain to me that they, the large clients in certain names were the only ones who truly understood the market, the actual market for certain securities, like their vision was so perfect that no one else in the market had the same information just by their sheer size, which was crazy.

And there's things you can do without even violating regulation. You know? Cause if you're a managing risk that think about managing risks, you have to manage risk. And of course you have perfect information. So think about the significant advantage that provides. So something like this across banks you know, it's interesting is that regulators might say You know, have you adopted this yet?

Are you part of this network yet? Because this network would enable you to act to sort of manage it more real-time so definitely that's definitely an interesting use case and it's, it's easy to see how that could be something more important, particularly for something like risk where you don't have to have that real-time information, but managing it night tonight and you know, who knows over time, maybe more real-time,

Brett: we walk then we run.

But also I think you bring up an important thing here, which. You know, secure multi-party computation. One of the things you can think of it is a way to cryptographically replace a trusted party. So if you had a regulator that sort of mandated that everybody give over their full balance sheet to the regulator could just run all of these tests internally.

And maybe under some conditions, they would actually do this. But the idea of a secure multi-party computation is now you don't need this trusted regulator. And so even if the trusted regulator exists and they say, we're going to do this once a quarter. The actual institutions might want to do it more than once a quarter.

They may want to do it once a day. Or they might want to do it under different hypothetical asset prices. They think maybe that regulator is being too conservative or who knows what. But they just want to do it more often or under different conditions. This kind of computation wouldn't replace the notion of as trusted regulator, but it would allow people to do more that the regulator wasn't in place to do for them.

Um, and so I think that's a thing that's really cool about it is that it's um, yeah, it would allow people to the sort of the consortium of banks or whatever the companies were interested in, something like this, to actually sort of a. To collaborate or to cooperate in a risk assessment that, that wasn't specifically mandated, but they think would be maybe more accurate.

Um, the other thing I want to point out is after in writing this paper, I spent a while looking at the Dodd-Frank bill, right? Dodd-Frank and the stress tests that requires are all local stress tests. They ask each bank individually to sort of run these kinds of factuals and they do not take into account any of the network dynamics.

So all of the like financial literature that tries to model. Contagion in interbank networks, right? None of that is captured in this kind of local stress testing. That's the type of stress testing that's done by Dodd-Frank or that's mandated by Dodd-Frank. Um, which is exactly the kind of contagion that you can capture by looking at the whole network, which you can do in a privacy preserving manner using a secure multi-party computation.

Eric: And, and I I've raised this point before this notion of you know, a regulator has access to a honeypot of information, even in the trading marketplace, this attempt to sort of aggregate data across asset classes, into a singular database that different exchanges can access that whole process of saying, if we give you all the information, now you can run your own analytics.

Well, that there's a lot of variation in security. Um, so you know what happens is you're creating greater risk by trying to contain it. You know, you're maybe not exceeding it, know, but if one party doesn't have the greatest security, you could have some real issues. When I was a general counsel of direct edge.

Um a lot of times the parties you think you trust to have the best system. Don't uh, NASDAQ had something called the director's desk where public companies would share information for years and that's not a regular they have a regulatory aspect in FINRA, but companies would share the most sensitive public information often relating to going public.

And it was disclosed to us, not to the general public that they had a a ongoing. For an unspecified period of time with access to an unspecified amount of information. In other words, it was significant and it was like we learned years. Uh, and you can just imagine you

know, somebody with a back door to this information, the way that they can abuse it and how it could have been used to the detriment of these public companies now, NASDAQ, it's a pretty significant organization.

Um why do we necessarily believe that somebody with an exchange license or any other party consuming it just because it's a bank or even the government themselves necessarily as a lock, they're a honeypot. Solar winds taught us one thing. It's that it's, this is not just you talking about deal department of defense, department of Homeland, all those institutions had potential vulnerabilities.

Do the solar winds hack. Anyway, it reduces honeypot, the regulators with that intermediary role, they should view that data as toxic and they should be looking, just say, Hey, how do I run red flag zero knowledge and minimize the participants access, including my own to this database. You know, in a way that's, un-encrypted let me leverage encryption, get the information.

If there's a red flag, then I collect the information. But before then maybe I,

Brett: don't a really important point that even if you have a trusted party who could aggregate the data and do the computation right now, they are a real target. And if you do this kind of distributed computation, since the data is never aggregated in one place there, there is no single party that you can attack.

Uh, and steal aggregate data. So that's I think, yeah, a really important point to make about this. The other thing that's a little tangential is, and this is actually another use case for MPC is actually sharing these, um security breaches. So this is companies have been very interested in this kind of thing, which is you have a bunch of companies that are maybe in the same space and they're all under constant cyber attack.

Um, and they would benefit on the whole of sharing essentially risk and breach information with each other, if they say attackers are trying to exploit this. And so we'd you better make sure you're all patched up in this, or I'm noticing some weird activity on, you know this type of server, is anybody else who's running these servers experiencing these same kind of, you know like penetration attempts for something like this.

And the issue is that in practice companies don't want to share any of this information with their competitors. Um, because. It basically makes them look that nobody wants to say, Hey this is how many cyber attacks I'm getting per day. Even if all, even if you happen to know all my competitors are getting the same number, like I still don't want to release that publicly, but you would benefit on the whole, if you could share that information and have some sort of aggregate database that just says you know, maybe that's looking for anomalies of saying there's a lot more people are noticing this type of threat recently.

So or just pushes out generic alerts based on this private sensitive data that each company has about their own you know, vulnerabilities and the threats that they've either successfully forwarded or been you know, vulnerable to in the past. Um, and so this is actually a great use case for MPC is sort of to get aggregate analytics on cyber threats.

Eric: And to just see what the costs were a little bit, and then we'll get into the main episode.

Brett: Yes.

Eric: Um, so the, you, when, in our discussion, you talked about the cheap version, like how do you get the benefits of encryption today without, with less cost? I'll let,

Brett: yeah so there's there's a bunch of different ways to do this.

One thing is you can use secure hardware, so secure multi-party computation and fully homomorphic encryption, or like software based solutions. Um, but you can use these trusted execution environments like Intel SGX, and they give you basically the full power of computing on encrypted data that you could get from fully homomorphic encryption, except it's all done in hardware.

And so it's just much, much faster. Um, and so this is a route that's like very practical today to do a lot of Pretty reasonable computations and the overhead is not very big that actually the overhead is that it's a little bit annoying to write code that runs an SDX. But when once you do that, you can with some tweaking get almost anything you want to run.

And there are more and more companies that are starting to come up that are writing essentially rappers that will try to take your code and sort of get it to run inside an SGX for you. So you don't have to think about the security side very much yourself. Um, and so this is uh, a great application and the big downside is that you're sort of relying on Intel to maintain the security of their SDX chips.

And there's just a whole slew of academic papers that say under certain circumstances you can really break the security of SGX. You know, it's still, nobody I think would claim that it's not better than nothing, right? Like it's absolutely better than nothing. And it's, it gives you some pretty strong guarantees that if I'm going to do some computation and use some platform like Azure or something, I can get them to run an Intel SGX enclave.

To me, for me, I can encrypt my data directly to the SGX enclave that data can be computed on inside SGX and the encrypted result can be returned to me. And I can get some guarantee that the you know, the system admins at Azure and everybody who's collocated on those machines really can't access my data up to the security of of SGX.

And this is like exactly the same kind of security I would get with fully homomorphic encryption, where I encrypt my data. I send it up, they compute on it and send it back except now you're just really relying very heavily on. Um, but we're seeing more and more applications of this because it is just orders of magnitude faster than MPC are fully homomorphic encryption.

Um,

Eric: and what are some of the outfits that maybe you're familiar with and you don't, I'm not asking for an endorsement that,

Brett: yeah. Um um, there's a company in Juna. Um, if you go to N juna.io, their goal is to basically make interfaces for you so that you can code in sort of to native languages.

You'd like to to actually get things to run inside of an SGX enclave. Um so yeah, that's one of these things. The other thing I want to say that's separate from this is, um well, actually back on the Google had a project Google, a silo, which was supposed to be rappers for this.

And I think that project has mostly died out, but it was a very cool idea. Um, that was for again, just to have standard, an abstraction layer that you could write in this sort of secure language and that could pile down to whatever secure hardware you happen to have. Um, um yeah, but there's more and more tools that are being developed for this.

Eric: Okay. So now we're going to, now we're going to talk about oblivious Ram. Now we talked a little bit about the concept of obliviousness. Um, but in talking about oblivious Ram can we start off with maybe drawing the distinction between circuit based versus Ram based you know, use cases

Brett: Or yes.

So one thing I was saying about when you're doing a computation with secure multi-party computation or FHE. The people who are doing this computation, they see the data movement, right? If I have fully homomorphic encryption and everything is encrypted, I still know when I'm adding two encrypted values or multiplying over when I'm writing one to a disk and reading it back.

And so even though the data is hidden, that what's called the control flow is not hidden. I can see what's happening again. You get this one of the clearest examples of this is something like a sorting algorithm. If I give you some encrypted data and I asked you to sort it if you did a traditional kind of sorting algorithm, if you could do a traditional sorting algorithm under FEG and then the sorting algorithm moved a couple of pieces of data and then stopped, that would leak the information that the data I gave you was like almost already.

Um, whereas if I gave you some information, if I gave something and told you to run this sorting algorithm under FEG and it moves stuff around and move stuff around for a long time, then you know, the data was not sorted that I gave you. And this, again, weeks, some information, which is in principle is a problem.

And there are actually a lot of applications where this leaked metadata, actually, it feels like maybe it wouldn't be so much, but there are cases where it's actually a big problem and allows you to infer a lot of information about the underlying data. So in whenever you're doing computations like this, you need to make sure that the computation you're doing is what's called data oblivious, that the control flow, that like data movement is independent of the underlying data.

So if you're going to sort them, things that the sequence of reads and writes and data moves is the same, no matter how the data came in to. Um, and this is in some sense, a separate topic from cryptography. There's a whole literature and algorithms about data oblivious algorithms. And you know, one of the main use cases is for privacy, but it's also useful in other situations where if you want to build hardware to do something the hardware is maybe not as flexible as the software.

And so you want a hardware circuit. That's always going to do the same thing. Um always have the same data movement. And so this one way to ensure that is with what's called the circuit model, where you have a circuit can be like an arithmatic circuit, which basically means that you have some data that comes in on wires and it gets joined in it gates when the gates are like addition gates or say, add the two things that came in and give me that put or multiplication gates where you say multiply the two things and give me the output.

And in principle, again, every computation can be represented as a. But that circuit may be very large because again, the circuit doesn't change, right? You're thinking of the circuit is laid out and no matter what date I put in the circuit the same sequence of additions and multiplications happens in this circuit, right?

You've seen it. It's like a sequence of pipes that are like joining things. Um, and so the circuits are great because they're automatically data oblivious. Um, and so when you're talking about secure multi-party computation, I'll say, I know you give me a circuit and then I will execute that circuit securely.

And the circuit has only a couple of different kinds of gates and it'll have an addition gate and the multiplication gate. And so if I can figure out how to do an, a secure addition and a secure multiplication and join them all together, then I can do a suckle secure circuit. And I happen to know that any computation you want can be expressed as a circuit.

And so now this is great. I can do any computation you want securely, but it doesn't mean I can do it efficiently. Um, because. I can't guarantee you that the circuit representation of

something is very small, right? The circuit, the initial step before the secure computation of taking what you want to do and converting it to a circuit, might be really big. And again, like one way to think about this is if you're writing a computer program and you have an if statement if this, then you do one thing and if you if something else, then you do something else, and this is like a branching branch in your program, in a circuit, right?

You can't branch like that. You have to execute both halves of the branch and then merge the results at the end. And this means that you don't get any of the benefit if I have a condition, which is like on rare occasions, I have to do some big computation. Um, but you 99% of the time, it's really fast.

I can't take advantage of that in the oblivious setting. And so I can't take advantage of that in this circuit. Because the circuit is always going to do to sort of the worst case amount of computation. Um, because it doesn't know, sometimes it's going to have to do that and it can't change its control flow based on the data that's coming in.

And so again, sorting is like this example I keep coming back to. Well, it's just if I have a sorting algorithm and if you think of a naive sorting algorithm, like index sort, right? If you were going to sort a deck of cards, what do you do? You look for the lowest card and you put it at the beginning, you looked for the next last card.

You put it at the beginning, right? The amount of data movements you do depends a huge amount on how sorted the data have sorted the cards where to begin with. Okay. If you try to do something like that as a circuit, you always are going to have to move the cards, the same amount, which means that you always have to move it the amount as if they came in like fully on shuffled.

And you can never take advantage of the fact that, Hey, look, it, that most of the time, the cards are already like, sort of partially shuffled. Um, and so this is already like a performance hit right away. That's right off the bat. Um, and it's a performance hit. That's really bad for examples. Um, like sorting and like searching.

So again, searching, what's the, the novice way. If I give you a deck of cards and I say find me the queen of hearts, right? You can just scan each card until you find it and then stop. But if you want to be data oblivious, you can't stop. You have to keep scanning each card all the way to the end every time.

So even if you find it on the fifth card, I still have to go through the whole sequence just to show you that I'm going to do that. Um, and. This data obliviousness means that your running time is always sort of the worst case running time. And so that's that can be very painful, basically, I, in terms of uh, of a cost.

Um, so the idea is you'd like to be able to do secure computations in what's called the Ram model, which is like the random access memory model. And so there's this notion of

what's called oblivious Ram and oblivious Ram is asking this question. You imagine you have, it was named, as you mentioned, you have a secure, trusted processor.

That's do you know, and you can think of that as yourself, or you can think of it as a really small hardware component. And this hardware component is going to write out to to your regular computer's Ram, which is untrusted, and it can hide the contents of the data by encryption. So I have some, I paid a lot of money for this really small trusted hardware component.

And it's doing all these computations for me, but it can't store all the information it needs. So it has to use the regular computer system memory. And it's going to encrypt everything before it puts it out there, but anybody's watching the system can see, Hey, here's the memory addresses it reads to and writes from.

And this sequence of memory accesses actually can reveal a lot of information about the underlying computation. So this is like metadata leakage. So even though all the actual data is encrypted, seeing where people read and write beaks, a lot of incoming. Um, and so the idea for oblivious Ram was can you make a compiler that takes a sequence of read-write accesses and makes it oblivious?

So that basically from the view of somebody who's monitoring your computer's memory, that all of the sequence of data accesses look the same, they're all look random and you can't distinguish, for example, a sequence where I read and write to the same address a lot. I can't distinguish that from a sequence where you would want to read and write to all different places.

And um, now, although the name is oblivious Ram, that the applications are much bigger than this right now. Um, cloud computing is a big thing, right? And you can think of this in the cloud perspective, instead of a trusted hardware component. I have me on my trusted computer and I want to outsource a lot of data to some database in the cloud.

And now I can encrypt all my data there, and I'm not even going to ask them to do any computation. I'm just going to ask them to store my data, but what do they see? They see all the times I read and write from a location in their database. So even in the simplest case where their database is just like a bigger Ray, it's just a list of all the encrypted files, they, they get to see every time I read from, if I read from the same file every time, or if I read from different files or the sequence of files I read from, or things I touch and don't touch, and this actually really leaks a lot of information. Um, and um, the idea of oblivious Ram again, is a sort of intermediate layer that says when you, it, on my side, when I say, I want to write to location three in this array, it actually randomizes things and writes to a different, random location on the actual untrusted memory.

And then when I want to read from something it reads from From, it keeps track sort of, of where that random location is, but it's constantly re randomizing and it's doing it in a really complicated way in such that I can actually prove that whatever sequence of reads and writes I was going to do, actually it gets a random sequence of reads and writes that's indistinguishable.

So if you, if the server operator who's operating the actual database and sees all the read-write patterns, they can't tell, for example, if I read and write from the same place every single time or every time I wrote from someplace read and wrote from someplace different. Um, and so this is the idea of oblivious Ram.

And so this is again, it's about hiding um, metadata access patterns. And there's a lot of really interesting work that shows like this kind of access pattern leakage is is a big problem. And there's, you there was a lot of works about things like symmetric, searchable, encryption, right?

Like you'd might like to have something where I put all of my emails in the cloud and I encrypt each one of them separately. And then I later want to search them based on keywords or something. And there's a lot of simple things that I could do here. I could just say, I'm going to encrypt all of my emails, each one separately. And then I'm going to encrypt like all the keywords from them as individual values and tag each encryption with like its key words. And if I do this with a deterministic encryption, so that every key word is encrypted. If I encrypted a second time and encrypts the same way, then later I could search it.

I could then like essentially each encrypted email has a set of encrypted, random tags with it. And when I want to search for Emails that have the encrypted economy in them. I will encrypt that and send it up to the server and say, Hey, send me back. All the encrypted emails that have this encrypted tag and everything stays encrypted.

And this sounds really good at first, that everything is always encrypted at all times, but the problem is right. The server sees re in principle, if you don't do this right, the server could see a lot of information. First it sees if like how many times I query for each individual word.

And if I query for a word, how many responses are returned, and knows if I, how many emails came back with this. And just these kinds of counts, statistics actually reveal a surprising amount of information. So just as a benchmark, we we downloaded the the old Enron data set. It's hard to get like good email data sets, but like just said, it's like a, as a hypothetical idea imagine you wanted to use this to encrypt emails and search for keywords.

And then we made a dictionary of like 10,000 keywords. And then we said imagine you have some sequence of keyword accesses, and all I give you is how many results are returned. So how often you search for a given keyword and how many results are returned for each for each query. Now, can you identify, can you just using this information, do you have like none of the underlying data, can you identify which keywords are being searched for?

And with basically like off the shelf techniques, we could identify like 20% of the searches which is like pretty big considering all the actual data's encrypted. This is just metadata leakage, and this is what you can do with just a little bit of results. We actually submitted this to a place and the reviewers are like, yeah, that's obvious.

Like this is not groundbreaking at all. And so we never ended up publishing it, but there's, there's a lot of really good work on this kind of inference that like inference is really strong. And if you do. If you set up a system where everything is encrypted, but the access patterns are leaked. You can actually reveal a lot of information.

Um, and so this is what oblivious Ram is getting at. It is how do you hide this access pattern information? Um, but then the kind of neat thing is that once you can hide this access pattern informing. Then you can imagine doing a more complicated, secure computation on top of it, where you're in, what's called the Ram model where you're doing computation.

That requires random access memory, where you're reading and writing to different places where it seems like the control flow would be different. But now the control flow is randomized essentially by this oblivious Ram. Um, and that means that you could then do secure multi-party computations that you don't have to first represent as a circuit that you can actually represent as a Ram program, which is how you would represent, um most of the computation that you would do insecurely.

So it gives you a sort of a more natural like programming environment with costs that are much lower. And so you can really reduce the cost of of doing a lot of these secure multi-party computations, if you could do them in the Ram model. So this is this paper you're talking about is trying to build what's called this display distributed oblivious Ram or DOE Ram.

And the idea of that is basically to use. Uh, secure multi-party computation, essentially in the Ram model,

Eric: draw some distinction where you see, oh, Ram being more efficient than MPC and where MPC is more efficient.

Brett: Oh, ran by itself is not doing any computation. It's just a way of interfacing to the sort of untrusted data storage, which you can think of as untrusted like memory card or an untrusted cloud server or something.

But it's like writing, reading, and writing to this untrusted array. And now you can use that as a building block to do more complicated computations that you'd want to do securely. Um, and so basically the, like the place where the circuit model shines is again in places where there's a move where the control flow would naturally depend on the data.

So just back to. Secure multi-party computation. Most secure multi-party computation is done in a circuit model where you say, first we represent the computation we want to do as a circuit, which means there's like a bunch of additions and multiplications say, and if you have a bunch of additions and multiplications, you can think of that.

What you're getting out is basically like a polynomial. If I add something a bunch of times we multiply it, sometimes it's like a polynomial. It's you know, X to the fifth plus five X plus seven or something, or it could be in two variables Y to the third times X squared plus. But right.

That's all you can do with additions and multiplications is make these kinds of polynomial. And we know, right? You can approximate everything with a polynomial or you can do basically everything with a complicated enough polynomial, but polynomials, sometimes the degree of the polynomial has to get really big, like polynomials don't naturally represent a lot of other kinds of functions.

And so on the mathematical side, if the function you want is sort of naturally represented as something like, like a polynomial, then it's fine to do it in a circuit model. And if it's not well-represented as a polynomial, then you want something that's in like the Ram model. And so I keep coming back to this example of searching and sorting.

These are examples of things that you can do much better if you have these kinds of other tools. And so then another type of thing that comes up here is like, why do you really want to do searching? Sorting? A lot of things are about like database joins as one question, right? So I have a data set with people's salaries and their social security numbers, and you have a data set of Social security numbers and their GPA.

And we want to analyze how does GPA re relate to salary? We want to actually do a database join on the social security number column, which requires some basic data movement to align these things together. And so this is like a building block of things that you would really want to be able to do.

Um, and this is hard to do in the like pure circuit model. You, you can do it. Um, and, but the, it, yeah, you get a, it opens up a lot of options. If you have this ability to sort of right. Um, Ram program. And it doesn't, it sounds like they're not mutually exclusive either.

Absolutely not.

Absolutely not. And that's how you would sort of imagined ideally what would go on is that people would have these secure multi-party computation compilers, which allows you to write what you want to do in some language. And it takes care of everything under the hood. And ideally the program would choose for you whether what it's compiling down to is like a circuit or a ramp program.

And this is sort of the goal we're getting to, right? Like for people to use this, we talked at the beginning about efficiency. Like one of the barriers to adoption is that these things are really inefficient. But the other thing is that it's sometimes hard to understand when, what we'd like to be able to do is say, look, you just tell me the computation you want to do.

Don't think about the security at all. You just say, look, imagine there's five people and they all have their private data. Tell me what is the computation you want to do and express that in some programming language that your. You know, comfortable with, and now all of this will get compiled to some secure multi-party computation where they're separate programs that each of these five people will run and they'll all communicate with each other in some encrypted way.

And at the end, the output will pop out. But you, as the sort of end-user, didn't have to really think a lot about what is the underlying MPC and. You know, what exactly is happening under the hood here. And so this is sort of the dream for all of this. And now this MPC compiler would really like to be able to have addicts disposal you know, an old Ram component that says, Hey, actually, like this kind of thing that you wrote is much more naturally represented as you know, in some kind of Ram program, as opposed to always sort of shoehorning everything into a circuit, which is what most of the things do now.

And so again, like in terms of shoehorning things into a circuit, right? If I have a program and I write a four loop and I say for equals one through five or something, but if the length of that four loop is itself, a variable. If I want to unroll that into a circuit to make it oblivious, I have to do the for-loop as many times as, as big as that variable could ever possibly be.

Because the number of times through that four loop, is observable. It's like everybody sees how many times I went through, even if the underlying data is encrypted. And so instead of having to unroll every possible Looper, every possible conditional statement, you would hope that you know, being able to have access to something that can do oblivious reason writes is would help with something like that.

Eric: Let's talk about the distinctions between three. Versus two party oh, Ram. And it seems to me like the three-party versus two party is something that exists independent of whether you're doing circuit based or Ram based. Um, but maybe expand on that, that

overall distinction in terms of the number of parties that can compute under a secure party multi-party computation model and the implications for.

Both data leakage and

Brett: efficiency. Yeah. So this is something right in a secure multi-party computation, you have multiple parties and most schemes are what are called threshold schemes that you do to find some threshold for privacy. So if I have five computation parties, I can set a threshold of three, which is that I need three of them to collude and sort of deviate from the protocol in order to break security, or I can set a threshold of four that I need four of them to collude or something like this.

Um, and the performance sometimes degrades with the you know, the security threshold, just like a trade off of security and performance. But there's also a question just how many people are involved in this computation. And in some cases there's naturally a bunch of people to begin with. So like in the example we gave about, you have a bunch of banks and they have their private balance sheets and they want to compute there's already.

Tens or 20 or hundreds of participants, but in some situations there may be very few actual participants who have data, right? If it's just me and you, and we want to compute there, isn't like some natural third party. And it turns out that for technical reasons, having three people who are willing to participate in the computation, actually you can do a lot of things that you couldn't do in the two party setting, or you can do them at least faster.

Um, and um the, the distributed oblivious ran the do-rag on paper that you're talking about. We worked in the three party setting because you could use all these different mathematical techniques and that's okay. You know, that's in some sense, almost a drawback because you would like to have something that works for any number of parties.

I don't want to say you need three people to be able to participate in this, but that's what we do need for this, because in order to gain the efficiency that we want, we did need three people and we couldn't get that same efficiency with with two players. Um, we'd like to be able to do that. And that's like an active area of research, but right now basically like the two party computation is slower and you can think of, um uh fully homomorphic encryption rate is a way there's principle.

There's only like really one trusted player. Like I had just right. I encrypt my data, I give it to the SunTrust cloud, they do something and they give it back. And so that's the real benefit of something like fully homomorphic encryption is that you really don't need sort of non-coding players at all.

It's just I encrypt my data, whereas I could do the same thing with a two-party computation but you if I do a two-party computation with the cloud than I'm actually doing a bunch of. Um, and the whole point of a cloud was I wanted them to do all the

work. And so if I want to get the same benefit of secure multi-party computation for like outsource computation, I would need two cloud servers, at least where the two cloud servers have to not be colluding.

I like secret my, share my data to the two cloud servers and they run a two-party secure multi-party computation or better yet. I have three different cloud servers and the three of them run this three-party secure multi-party computation because that can be really fast. And right now with current technology, we can do that faster than we could do with FEG, but it's annoying to say, we need three different non-polluting cloud servers to be able to run this.

Um, like basically, we'd like to be able to do things with sort of as few parties as possible. Um, just because it gives you more flexibility. If you can do a few parties, you can always do it with more. Um, and in some cases though, having more allows you to increase efficiency. Um, and usually the big efficiency gain becomes between two and three.

Um, and you, there's an efficiency loss. I was saying at the beginning that there's efficiency loss as you go up because the amount of messages back and forth, you're sending messages to everybody. So the number of messages grows with N squared where N is the number of parties. So you really don't want like a hundred.

It's not like the more people you got, the more efficient it is, but good. Moving from two to three, it opens up a whole realm of sort of algorithmic techniques that do tend to increase efficiency. So right now we do see most of the three-party protocols are still a little faster than the.

Eric: And so like in the context of financial institutions, just to spell this out a little bit more we could have a hundred participants in the network, the notion of secure multi-party computation isn't that you have a hundred participants performing the computation because to your point, that's just, it's just a mess.

So ultimately you have to choose at the outset it's part of the architecture say which are the three parties that can do this either external or even potentially part of the

Brett: network. It's usually how it's done that. You said outsourced MPC. We have a hundred people who are going to submit data and we're going to choose three computation servers and they, again, they can be the three biggest or most powerful of us in the group where they can be three external parties.

And we're all going to secret share to them in the security we get. The guarantee we get is that if those three people do not collude with each other, then we're guaranteed that all of our underlying today is going to be secure. Um, and so this is a nice model because then we don't have the sort of blow up in communication that we'd get.

If we all tried to participate in this secure multi-party computation and these the three year, the five computational servers, however many you choose can be really dedicated. You know, they can actually, you can spend more money on hardware and have them have really good internet connections and things like this, which you may be couldn't guarantee if you wanted 150.

All participating in it, but you get the best security if secure multi-party computation in principle would allow every one of the a hundred banks to participate in this, each one run their own node. Um, and that would be the, sort of the best security, because then you'd say you'd need, you could have it that all the other banks would need to collude against you to break your privacy or something like this.

But from an efficiency standpoint, usually we don't want to do this. And so we take a look again, a little bit of a hit on privacy, which is to say now that the security guarantee is that these three servers would have to collude, or these five servers would have to collude instead of 50 out of a hundred of the other servers.

And somehow it seems more plausible that three out of five would collude than 50 out of a hundred. Um, and so you have this slight hidden privacy but you got a major gain in efficiency. And so this is like the kind of outsource MPC. That's done a lot. And one of the things I think though that's interesting in this context is that you can still, you can get back a little bit of this security by enforcing the NPC clients to run inside of SGX.

So we said before, if you have SGX, you could just throw away all of MPC and just say, just run everything inside of an SGX, but then you're relying on the security of Intel, which in some cases I think is fine, but in some cases, maybe you're worried about that. But now if you have an MPC and you have three.

You do a three-party MPC, but what if those three NPC clients were all running inside their own SGX enclave, then in order to get any one of them to misbehave, you'd have to break that SGX enclave to get it, to violate the MPC protocol. So now your security is not just that these three parties got together to defraud you it's that these three parties got together and they all individually broke their SGX in order to get their software to be fraud.

You, and now that's just like a whole new layer of security on top of that. And so that gives you yeah. Oh, sort of you got the kind of layered security benefits of the kind of MPC and SGX on top of each other. So I think that's really nice. And I think that's what the avalanche bridge is doing.

They have a four party, um like multisig, basically, which a threshold signature scheme which would need three out of four. You can think of it as like a very simple MPC that needs three out of four corruptions to steal money out of the bridge. But if each of those

four participants are running inside of an SGX, now the guarantee is not that three out of four of those people need to collude to defraud you.

It's a three out of four of those people need to agree to collude to defraud you. And then they have to figure out how to break their SGX and then they can try to steal money from the bridge. And that's again, like a much harder. Yeah. So SGX again, down and dirty way to accomplish it, particularly with the wrappers.

Eric: Uh, you can build on top of that. Um, to, to, to shift gears for the last part of our episode here thought we talk a little bit about you know in the context of AML, at least one of the things that's come up a lot recently more so because of all the FATF guidelines and such is how do you build a, um it could also apply to traditional finance.

How do you build a trustless environment where people don't have to necessarily reveal a ton of personal information about themselves, but yet through white listing and associated transactions with white listing, we can start. Not only determine who is, who is not a who's on the black list, but also black malicious actors can obviously take over whitelisted accounts often happens and it and still act through other accounts that aren't on the black list yet.

So obviously it involves a lot more transactional analysis. Um, Brett and I were talking about both the use of credential management and zero knowledge proofs and how they could be utilized more effectively to facilitate. You know more on the KYC AML site in a way that doesn't necessarily reveal the identity of the participants.

Brett: Yeah. Um, you know in addition to secure multi-party computation, one of my big areas of focus right now is in blockchain space and with a emphasis on privacy, but actually a lot of different aspects of blockchain. But one of the ideas of block, one of the blockchain use case that I'm really excited about is this kind of credential management.

And so the idea would be that you have credential issuers, which you can think of, like the DMV will issue you a driver's license. And the w that would happen by they give you a driver's license and they digitally sign it. Or you get a digitally signed passport from the passport authority, or you get a digitally signed diploma from your university or something.

And that you wouldn't put these documents on the blockchain because most of our blockchains are public. And there's no reason for them to go on the blockchain that your credential issuer would give them directly to you. And when you wanted to use those credentials to somebody else, you could present those credentials to somebody.

So you want to get into a bar and you need to present your driver's license to show that you're over 21, you present this digital credential and they can verify the signature that it

came from the DMV, but the issue there is. So at this point we have no blockchain that could somebody issued you a credential and they verified it.

But the issue is about this key management. In order for somebody to verify your credential, they need to know what is the public key of the issuer, right? Because if they have the wrong public key, if they had the wrong verification key, then I can forge credentials. If they think my key is the verification key for the for these credentials, then I can issue credentials to anybody that I want.

So they need to know what is the the true verification key for the DMV or for the password authority or for the university that's issuing credentials. And information goes on the blockchain so everybody can see, Hey, this is the public key that's associated with the Pennsylvania DMV. And then the other thing is that when you do this kind of credentials, where you issue a signature on something, you might need to revoke these, right? If I get a DUI or something, then they want to revoke my driver's license, but you can't claw back data. When somebody gives me a signed document, they can't ever pull it back blockchain or no. All they can do is say, Hey, look this, the signature.

I want to take that back. And so there needs to be like a revocation list and some information about that can actually also go on the blockchain. So they the DMV can post a revocation that says, we re revoked this signature hash. And so now when I present my driver's license, They can look up what is the DMVs I address on the blockchain and they can verify it.

And then they can look at the set of revocations that's also on the blockchain. And so in that sense, it's also permanent and I can't easily tamper with it to see if the hash of my driver's license was one of those ones that was revoked. And so again, I can have some privacy here that nobody needs to know.

It doesn't say like Brett's, driver's license was revoked, but when you go to check this value, you can see that it was revoked. And so you can maintain a lot of privacy this way. You can have a lot of credentials being issued by all different credentialing authorities. And the blockchain is just a coordination mechanisms that people know the public use of these different things.

And then on top of that, you can now start to layer all kinds of really interesting zero knowledge proofs for things like when I go to vote, maybe I need my driver's license to prove that I'm a resident. Um, but I, my driver's license has all kinds of other things. It has my photo and is my eye color, and it has my height and weight and my date of birth and stuff.

Um, and maybe I don't want to reveal all that to the people who are running the voting station, I just kind want to prove to them that I'm a resident and that I'm over 18. And I say,

none at all. The rest of is none of your business. And so what I'd like to be able to do is issue a zero knowledge proof that says I have a driver's license that was signed by the Pennsylvania DMV.

It says my date of birth was more than 18 years ago and that I am a resident in this. And I want to prove that in zero knowledge and that they can verify that statement. And so if you can do that and that's actually a very, a fairly simple zero-knowledge proof. Um, now we can have really nice sort of privacy preserving credentials that I can reveal only kind of the amount of information that you need to do, the things that you want to do.

Um, and so I think this is like a great blockchain use case it's it's very hard to get people on board to do this. I can't imagine the amount of work that would be needed to get the DMVs from all different states to actually adopt a system like this. But in principle this type of credentialing would be really useful.

And you could do it for this kind of KYC or AML to say I have a signed documents that say I went through KYC with somebody. And now it doesn't even have to be sort of these like legally binding documents, like a a driver's license, like it can be, if I go through KYC with somebody and they issue a signed thing saying you know, I checked bread out and he's reasonable.

Now somebody else can choose to accept that or not. And they say, you know look, you went through KYC with this. Person, I trust them. So I'm willing to accept you onto my system. Um, and so you can use it again as these kind of um access cards or credentials for other things that are opt in.

And you can imagine that being a really good way to sort of maintain some of the privacy and anonymity of the blockchain, but still have certain parts of it, the accessible through some kind of KYC. And ideally again, you'd have that KYC be reusable. Like it's really annoying if every different service I want to provide, I have to scan my driver's license and send it up and give like a utility bill and whatever else I have to do.

I want to do that once and have one authority say we're issuing you this like credential. And then everybody else trusting that authority saying like their KYC was good enough. So it basically, it basically it, it creates potentially a honeypot and the issuing authority, but this is the fact of the matter is I think there's sort of a vision of maybe two worlds prospectively as a release, AML KYC.

Eric: There's the one side that envisions that. You know, an identity will need to be revealed at some point in order to do the underlying KYC. And then there could be a real utilization of it, but it doesn't have to be continuously reutilized like today. How many times have you had to show your past send your passport online?

It's every time I have to do that in the real world, it's this is ridiculous. You know, how many inference of my passport and driver's license are floating around just for multiple requests.

Brett: It's a terrible thing for a user privacy. I hate having to scan my driver's license to send it when somebody and say, I have no idea what you're going to do with it.

Whereas if I can see just issue a proof to you that I have a passport, right? You can't that, that leaks no information about me. You can't sell that. You can't misplace it. Um, and so, so that's really nice, but then there's also like there's the user privacy thing, but then there's also this thing about reusability is also really nice that I don't have to keep sending things to all different.

Eric: And then there's the other vision, which is you never, after her, you never have to reveal anything ever because it's based on blacklisting addresses and it's based on associating blacklisted addresses based on transactions with other white lists and setting thresholds and all that.

So it, it's definitely two competing visions of and maybe ultimately

Brett: we ended up with both. So I think blacklisting addresses does not seem very useful to me in the sense if there's really no underlying. Um if there's no underlying identification mechanisms. So like just looking at, you know like the stable coin contracts you know, tethers, USD, T or S um, circles.

USB-C like, if you look at them on the Ethereum blockchain, you can blacklist addresses and they both have blacklisted, a whole bunch of addresses saying these addresses are no longer able to move their underlying stable. And shockingly, this has been somewhat successful. There's money when they blacklisted these accounts, there was money in these accounts.

But if criminals were sophisticated you should always be able to outrun this blacklist, right? Just start another, spin up. Another address, spin up another address. And all you have to do is issue a transfer from your account to this new address before the blacklist address comes in. And right, like right now on Ethereum, we have this problem of like maximum extractable value in this MEV.

And you can think of it as like a front running thing. So even if circle is really on top of things and they're monitoring things right away, and they see that this address stole some USBC and they say we're going to blacklist it. And they are just like on it, which is hard to imagine that they can be so fast.

But even if you imagine. All you have to do is get your transfer in before theirs is in. Right. But there's also consider this that's level one, but level two says, and again, we can hit it from the perspective of either. You can only trade with blacklisted or you can't treat a

blacklisted accounts, or you can only trade with waitlisted accounts, which is another threshold to achieve white listing, but a black list of the account that transfers assets to a waitlist of the account that could potentially gray list or blacklist that white list, that account and so on and so forth. But even that is so hard to track because there's all this pooling happening. So it's true that if I steal some USBC and I think they're going to blacklist me and I just immediately transfer it to another address they'll blacklist my note new address, and we can play this cat and mouse game.

And I think if I were a sophisticated criminal, I would imagine that I could front run the blacklisting transactions when circle sends a transaction to the blockchain that says you know, police blacklists distress. I should be able to basically get my transaction to transfer in first, but I wouldn't transfer to a normal address.

Because they'd blacklist that I, what I do is I go and I tell. And I take my USBC and I drop it in the dyes and make her dad died, peg stability module. I immediately transfer it for di and now, I have all this dye and the dye cannot be blacklisted. The dye contract is not allowed that to be blacklisted.

And now all of my, the USBC that I stole is stuck in the dyes peg stability module contract. And it's pooled with everybody else's and if circle blacklists, the whole di peg stability module, that's a huge thing. I actually just looked at dyes tech stability module is the number one holder of USB-C on youth, right on Ethereum right now.

Um, and right, if I steal a couple million dollars worth of USB-C, they're not going to blacklist this whole contract because of it. Now they can claw back, they can actually, without blacklisting, they can claw the funds back. So I deposit $30 million into the pig stability module. Um, and now they claw back.

And they can call back that $30 million, but that doesn't hurt me directly. That hurts all the Dai holders equally. It doesn't hurt it. Didn't claw back. My dime. My dye is not tied to the specific USBC that I put in. It's just backed by the sort of global amount of collateral in all of, um in all of the maker ecosystem, not just what's in the peg stability module.

So once I convert from USBC to die, there's nothing that you can do to me to pretty much. And this is not I don't want to single out maker. You can do all kinds of things, like if I stole USB-C on Ethereum, what do I do? I send it over the avalanche bridge on to the avalanche network.

And what do they give me on the other side, they give me wrapped USBC on the other side. And that wrapped USBC is not controlled by circle. And so that can't be blacklisted by circle. And my USB-C is stuck in the avalanche bridges contract. And what can they do? They're gonna blacklist the entire avalanche bridge.

Like I don't think so. Um, and if they, again, if they claw back from the avalanche bridge, it doesn't claw back. The USBC that was issued on avalanche, it just sort hurts the whole pool equally. And these kind of blacklisting of like individual tokens is hard for me to imagine them actually working.

Um,

Eric: but interesting. There, there is actually moves to not only IP addresses, but I think like a, and I don't know too much about it yet. Uh, it's like an AOP in Switzerland where you have to prove your wallet ownership in order to onboard at any exchange or anything like that. So it basically, it's bringing it down that other

Brett: level.

And so then this is like a white listing thing. And so white listing can work. It's just extremely onerous, right? But from a security standpoint, it sort of works. If you believe that your white listing process is good then. Yeah. Um, that's the

Eric: credential management is effectively a waitlist, right?

Brett: Yeah. Yeah. Yeah. Yeah. Just depends, but again it's very hard because these things are so interoperable, even if I white list certain things how do you handle pools, right? So you say, I can only interact with I don't even know how you would do this. If I say I only interact with, see if I'm on a white list.

Does that mean you do have to put right. What about all the DFI? Do you white list, like the curves three pool, which is like a huge holder of USB-C, right? Or you know, do white list maker, Dao peg stability module, or do you make white lists? One of these bridges. And if you don't right.

That's like a huge blocker, but if you do, then it people can use that as a way around the white list. Cause they're effectively holding this thing or it in some of those it breaks things in weird ways, which is if I am a. You know, if I could only use USB-C white listed, but I have some LP tokens for the curve, three pool, which entitles me to withdraw from the curve three pool.

But USBC says, Hey, we're not going to let you withdraw the USBC from this pool. Then these like LP tokens are worth less to me than they'd be worth to somebody else who is white listed. And it, it becomes very complicated and it's not like I would think you could still probably get around that with some thought by just again, using this kind of derivative or wrapped token, right?

Like a deposit into some pool that's white listed. And now I have that LP token, which entitles me to a share of the school. Um, and if somebody on the white list can withdraw, I

can't withdraw, but that's okay with me. I can still trade it around knowing that it is valuable to some people on the wait list.

You can withdraw. No, it sounds like credential, whether it's credential management or white listing, it seems like it's a similar issue as it relates to taking part in the pool. Um, I guess we won't solve it on this podcast episode, huh? Yeah. Well, great. We'll definitely include links to that as well in the show notes okay.

Eric: So Brett, thanks so much for coming on. This was great. Yeah. And look what we'll call it there.

Brett: All right. Thanks. .