TADGE DRYJA: Today we're going to talk about wallets and SPV. And if you don't know what SPV stands for, that will be defined as well, so don't worry.

First, you get your software. That's something of a problem, like how do you know you got the right Bitcoin software, there can be issues there, but anyway-- there are multiple implementations that should be consensus compatible. I often use BTCD, which is a version written in Go, but the main implementation is written in c++.

So, you get the software, and then you somehow connect to peers in the network, but there's all these asterisks, which mean it's not completely decentralized. You have to find where to get it, you have to find who to connect to.

Once you do that, you get all the headers, which are 80 bytes each, you verify all the work, and then you start getting the blocks, looking through the transactions. You replay the history of the last nine years of the coin, and then you arrive at a UTXO set, an unspent transaction output set, and it should be the same as everyone else got. Everyone else has the same headers, the same work, the same transactions. They'll get the same set of which coins are encumbered by which keys as you do. And the idea is it would be very expensive to have different UTXO sets because you'd have to do all that work.

So that's how a node works. But what about dealing with actual money? So far, we've just looked at here's how to get your node running, here's how to observe the network and come to the same conclusion as everyone else, but you probably want to actually do something with this. You want to pay people or get paid. Those are the two fundamental functions that this tries to address.

So the software that manages this feature is called a wallet, and it's not necessarily the same software as what's connecting to the network, downloading, and verifying. In the case of Bitcoin Core, it is, although many of the programmers of Bitcoin Core wish that it weren't. And there's sort of a long-term goal of, it'd be really great if we could pull these two things apart

into maybe separate binaries, separate programs, something. But they're really intertwined, and it's kind of ugly. But there are other things that are separate

OK, so wallet software functionality-- seems simple, you send and receive money. Simple. Of course, you need to receive money before you can send it, so let's start with that.

OK, so we did not talk about receive addresses. We did talk about the script, and how it's generally used pay to pubkey hash, where you put the hash of your pubkey in your output script, and then in your redeem script, in your input, you put the pubkey itself, which is then checked against the hash, and then the signature is verified.

Most likely, if you've looked at Bitcoin at all, you've seen these types of addresses. They usually start with 1, they're a bunch of characters long, a weird mix of lowercase, uppercase, Latin numbers, and letters. There is a standard for converting a 20-byte pubkey hash into this address. So the idea is since almost everything is using the same pubkey hash script, you can forget about the opcodes, like op_dup, op_hash160, op_equalverify because they're always the same. And so it's this standard, like, OK we're just taking that 20-byte hash, and now let's convert it to something hopefully somewhat human readable and writeable so that people can write it down, say it over the phone.

This is the one Satoshi made. It's got 58 characters and then the last 4 bytes, which ends up being like 5 or 6 of the letters, is sort of a checksum, where you take the hash of the first however many letters and then that's supposed to like equal to the next one. So, hopefully, if you typed something wrong, it doesn't change the hash and then you send it to the wrong place and then no one has a matching key for that.

There's a newer standard for this where all the all the letters are lowercase. That's introduced, actually, today in Bitcoin Core. Version 0.16 came out, and so there's a new standard called Bech32. They did some research, and they found it was actually much faster to transmit over the phone via voice because you didn't have to say whether things were uppercase or lowercase, which ended up being very annoying for people, because, you know, 1 big F, a little f, 1, 2, big E, 4-- it's annoying.

Anyway, the idea is this is just an encoding of the 20-byte pubkey hash, so when you type this into a program, it reads this, converts it into a 20-byte hash, builds the output script. OK, so the outputs are all the same. So this is sort of like a UI thing. The addresses don't really exist

at the protocol level. Any questions about addresses? OK. We're not going to put-- UI and usability is super important, but not the focus yet of what we're doing.

The idea, in a lot of cases, is you want to receive money and know that you received it, or somehow interact with people over computers, and you could put a bunch of addresses on a server, but keep your private keys offline. Because if you keep both your public key and your private key on the same computer, that's kind of an attractive target for someone to break into your system because they say, oh, this guy's running Bitcoin and he's accepting payments. There might be a bunch of money in this computer if I can get into it. I can take all the money.

So one issue that people ran up against pretty early is-- well, let's say I generate 10 keys-- 10 private keys, 10 public keys, 10 addresses. I put them on the server, then I run out. And I can reuse addresses, but that can hurt privacy because people can then see that the same people are using these keys.

So is there any clever way we can generate pubkeys without the private key? Is there, given all the fun key stuff we've talked about, can anyone think of any clever ways to do that? OK, well pretty straightforward.

This is called BIP32, Bitcoin Improvement Proposal 32. This is a super-simplified version, but this is the basic idea of what they do, and they do it much more involved and complicated. But basically, you've got your public key P-- big P-- and some kind of randomized data-- randomizer data, r and your private key is just little p.

So the idea is you want to send to an address, you want to generate a new address. Well, it's just your public key plus the hash of r concatenated with 1 times G. And if you wanted to make this 2, 3, you can make this any number you want.

And then your private key is just going to be that same public key plus the hash of r. So you give someone some extra data, which they can throw into a hash function. Use this as a known private key and you add it to your private key. So no one just knowing this data can spend from it. That's really nice because then the server can generate arbitrary numbers. Does this make sense?

AUDIENCE: What's the difference big A and a?

TADGE DRYJA: Oh, yes. So in the last one, big A is a public key. It's a point on the curve. Little a is the private key. I screwed that up. That does not have a G. So G is the generator for the group. G is how

you convert from a private key to a public key. You just multiply by G, which is just an arbitrary point. Yes.

AUDIENCE: So your private key doesn't change?

TADGE DRYJA: So in this case, there's two private keys. There's your standard private key that you actually just randomly created-- this number p, multiply it by G, to get big P. But your private key for any particular address does change. You're adding the hash of r, 1 or r, 2, r,3. Yes?

AUDIENCE: Assuming the size of r is relatively small compared to p because don't you have to keep track of the nonce?

TADGE DRYJA: R should be, like, 32 bytes or something. You know, you don't want any--

AUDIENCE: Do you have to start with it every time that you create a new hash code?

TADGE DRYJA: You sort of don't. What you can do is, you have your server. You say, hey, I'm going to accept payments for cookies or shoes or whatever I'm selling. And then you give the server your public key, P, and the randomizer r, and you just say to the server, go wild, make whatever number you want here.

This number should be fairly small, let's say less than a billion. And then when you want to find how much you've gotten paid, well, you can generate a billion private keys here-- some reasonable number that the computer can actually do it-- and you could just increment this, generate a ton of addresses, and look for them on the blockchain. So does that makes sense at all?

BIP32 is actually quite a bit more involved and complicated. This is the basic idea, but they make trees of it, and you can say, oh, well, we can make instead of just one level, we can make a whole tree of these things, and have different accounts, and make a really full-featured system in case people want to use this kind of thing.

So you can put the public key and this random data on the server. The server can make addresses as needed, really quickly. And what's nice is observers can't link these addresses. If you're just looking at the blockchain, you won't see the difference between a sub 1 and a sub 2, if this number and 1 and 2 because it's going through this hash function, you never see that.

To an observer, it looks like all completely different addresses. And if someone hacks into the

server and finds this P point, and this r randomizer, well, that will allow them to link everything. Now they can see-- oh, we can also generate all these addresses. We can see that it's all the same person. But that doesn't let them steal any of the funds. So compromising the server with this, well, you lose the privacy, but you don't lose your money, so that's a pretty good trade. Other questions about BIP32? So that's one of the features for wallets. You've got to do this.

The basic procedure-- you're going to request a payment. And you're going to say, hey, if you want this jacket, send one coin to address F8f12E. And it's sort of important to note that Bitcoin doesn't solve the problem of people paying money and not getting what they paid for. That's out of the scope of this, although there's a lot of people that say it should, but know it doesn't do fraud protection. It's like, hey, I gave you the coin, you didn't give me the jacket. Well, Bitcoin worked fine. Bitcoin made the coin move. That's Bitcoin's job. The fact that FedEx never delivered your jacket, well, that's FedEx or the retailer or all sorts of things like that.

You know, I don't want to say it's not a problem. It certainly is, but it is seen as out of scope. It's, like, you know-- this is a money system. This is money. Your dollar bills don't ensure that you're getting what you paid for. That said, there's all sorts of things that do this kind of thing, and do try to ensure delivery versus payment-- atomic swaps, HTLCs that we'll talk about later, Zero-Knowledge Contingent Proof-- all these different things do sort of work on top of Bitcoin to try to help these kinds of things.

In practice, though, if you're actually buying a physical jacket that someone's going to deliver to you, there's not really a good cryptographic proof of jacket delivery. So some reputation is involved.

Then, from the merchant's perspective-- so, I sell jackets. I want to know if someone paid me. I have something on the website, put in your address, now pay this address. So you add all your pubkey hashes to a big list in your software. You say, OK, here's all the addresses I've created. They're all these 20-byte pubkey hashes. I put them in a map or some kind of array or some database, whatever. And from then on, every transaction I see on the network, I also look at the output scripts.

So before, when I was verifying the blockchain, I actually never had to look at the output scripts until they got spent. So when I was downloading transactions, I would look at their signatures and look at the old UTXOs in the UTXO set and match them up and verify. But

where the money got sent in these new transactions, I didn't really care. It could have been sending it to zero. It could have been sending it to some weird address that probably was wrong.

There was no-- and I believe to this day, there's still no output validation that happens in Bitcoin as a consensus rule because it's not important. Where are you sending the money? Well, wherever you want. And are you able to spend it? We'll deal with that later. If you want to destroy your money, fine. I'm not going to look at the output. There's no invalid output. There can be an invalid input, which-- there can be an output which you can never actually use, but you're free to send to it. So that's sort of one of the rules in Bitcoin.

However, when we're actually looking at our own money with the wallet, we do look at the output scripts, mainly to say, hey, is this ours? Are we getting paid with this transaction? So you look at every output script, and if you see one that matches, hey, we got paid. So you see a transaction, one of the outputs-- hey, look, that 20 bytes-- that's me. Cool. That's one of the addresses that I have, let me keep track of this. This is now money that's in my wallet.

So you keep track of the received payments, you save them to disk in a similar way to your addresses. You use some kind of database or map or something, something efficient. And then you don't need to save too much information. You need to save the outpoint, the txid of the transaction, the index. You probably want to save how much, your amount, and which key it was, so the actual 20-byte pubkey hash. You can look through all your keys, but it might be nice-- oh, that's my 17th key that I've saved in my database or something.

You may also want to save the height information, when it got confirmed. So we're not going to talk too much about unconfirmed versus confirmed today, but this can be an issue if you see a transaction on the network that's not yet in a block and it pays you. You're like, hey, I got money, but it's not confirmed yet, so have I really gotten money?

I am able to use that output to spend somewhere else, but now I've got two change transactions, neither of which is confirmed. And now the second one can only be confirmed if the first one is, so that can get ugly kind of quick.

For simplicity's sake, let's just say that you wait until it's in a block before your wallet recognizes it. Most wallets do not do that, but most wallets maybe should. There can be a lot of weird attacks where you say, oh, I got money, and then since it never really was confirmed at all, it's pretty easy for someone to double spend.

The whole point of the Bitcoin system was you can't double spend because it's got all this proof of work on top of it. It's in a block. But if we show in the UI, hey you got a payment that has not yet gone into a block, well, there's no assurance that it won't be double spent yet, because it's not in the blockchain. But most wallets will show that and usually they'll make it in like red, or put a little exclamation point or something to try to indicate, hey, this is unconfirmed, but that doesn't always get across to people. So it may be safer to just not show it at all until it's in the block.

OK, so you amass all these you UTXOs, you're running your node, you've got all these addresses you've given out to people, and then every transaction that comes in you look-- hey, do any of these pay me? And sometimes you'll find one that does, which is great. And then you save that to your disk and, great. Now, next, I want to spend them.

OK, any questions about the getting money procedure? Yes.

AUDIENCE: So, what's the height again?

TADGE DRYJA: The height is what block it's in, so height zero is the first block, zero block, and we're now at height 500,000. Usually, in diagrams, it goes this way, but I guess it could go up in numbers, I don't know.

Yeah, so next, you need to spend them. Spending makes sense, right? It's not too crazy. Let's say you want to send six coins to someone. So what you do is, you look through your set of UTXOs that are your UTXOs, and you try to find some such that the total number of coins is over six, and then you use them as inputs, and then you add your outputs.

So, for example, I've got two UTXOs. These are much smaller numbers, but this one has five coins in it and this one has three coins. So I received, at two different times, once I got five coins for a fancy jacket, once I got three coins for a less fancy jacket. And now I want to buy something, and I want to send it to Bob, and I want to send six coins.

Well, I've got eight coins in my inputs, so I'll send him six coins. There's two coins leftover. If I don't have this, it just goes to the miners, so I create my new output, which is my change output. And then I send the remainder, two coins here.

Now, if it's actually these nice, round numbers, the fee would be zero and it would probably not get confirmed on the Bitcoin network. You do have to have a small fee right now. It's really

small, like a couple of cents now. It was pretty high a few months ago. But this will work. This is the basic idea.

So, you look through your UTXOs, find some, OK, output six, output two, sign them. Once you've built this, sign it all, broadcast to the network. Make sense? Yes.

AUDIENCE: Is the change UTXO, like, your own?

TADGE DRYJA: Yep. Yep. Generally, what you'll do is, you'll make a new private key, calculate the public key, hash it, make this address, and then add it in here, all automatically.

You can, and some software does, just only use one key and one address, and so it'll be pretty clear because the keys for this signature will be the same as this key that it's sending to, and so then it's really clear. Even without doing that, it's usually pretty clear and people can sort of guess, well, either Alice is sending six coins back to herself and two coins to Bob, or she's sending six coins to Bob and two back to herself. Or maybe six coins to one person, two coins to someone else entirely. That's pretty unlikely.

Usually, metrics will try to analyze the transaction graph, and say, oh, the smaller outputs are usually the payments and the larger ones are the change, but you don't know if the addresses are all different. Yes?

AUDIENCE: How does the fee get paid again?

TADGE DRYJA: The fee is the difference between the inputs amounts and the output amounts. So, in this case, the fee is zero because I got 5, 3, 8, and then 8 here. So, really, what you do in real life is, you'd make this 1.999 and then the fee would be that 0.001, or whatever that the miner gets in the coinbase transaction.

That's another way to try to identify change outputs. If you actually had 5, 3, 6, and 1.999, I bet the 1.999 is going back to yourself. Nice, even, round numbers seem like real payments. And then if you've got one bunch of nines at the end, oh, that was probably just reduced a little to make the fee.

But these are all guesses. If you're a third party observer looking at a transaction, you don't know. This could be two different people, or this could be an exchange and it's hard to tell, but you can get some pretty good guesses.

Yes?

AUDIENCE: In terms of fees, so if you have no fee or you had a really small fee and buyers are requiring something higher, that just sits on everyone's computer. They still share with each other or do they sit there until maybe there's a block?

TADGE DRYJA: There are multiple thresholds. So, there's the relay threshold, which right now I believe Bitcoin is one Satoshi per byte. So, I think we said Satoshis are the smallest unit possible in Bitcoin. So one coin is actually 100 million Satoshis-- and there's no decimal places, that's just a UI thing.

So right now, the minimum relay fee, by default, is 1 Satoshi per byte. So for a 250-byte transaction, you need 250 Satoshis, which is some fraction of a cent-- maybe more than a cent now, I'm not sure. And then the idea is if you see a transaction below that, you won't even relay it to anyone else. You'll be like, this is so cheap that I'm not going to bother sending it to everyone else. I'm just going to ignore it.

But above that 1 Satoshi threshold, you will accept it, verify the signatures, and then pass it on to everyone else you're connected to. But that doesn't necessarily mean it will get into a block anytime soon. It's actually been really interesting the last, I'd say six months, where the fees went up enormously and you see this really crazy price inelasticity, where people who were paying one cent then started paying 10 cents, a dollar, $10, $20.

And what's also sort of optimistic-- that made me feel good-- it's like, well, clearly they were getting 20 bucks worth of utility out of this because they're now perfectly willing to pay 20 bucks and they were paying one cent a few weeks ago. That's kind of weird. And then it's now gone back down to, like, one cent. But it's very inelastic in that there's a fixed size for how many transactions you can have per minute or per hour and when people really want to get in there, they have to just bid everyone else out.

So we'll talk about the fee markets in a few weeks, and replace by fee. That's sort of a new evolving thing, but it's been really interesting to see in the last few months how it's changed.

Yeah.

AUDIENCE: At $10,000 per Bitcoin, a Satoshi is 0.01 cent, a hundredth--

TADGE DRYJA: Tenth of a cent.

AUDIENCE: Hundredth of a cent.

TADGE DRYJA: Hundredth of a cent. OK. So, the minimum relay fee would be more like 2 and 1/2 cents. So that's-- you know, it's not zero. That's still-- a fee, right? And you get enough of those and you start making money.

But it's also interesting recently, it used to be that the initial new coins coming out to miners was just overwhelmingly the majority of what they had earned and people would ignore fees as a miner. But then, in December, January, I believe miners made more money in fees than in new coins being generated. I'm not sure if that averages out. There were definitely weeks where that was the case, or at least days, I'm not sure if it's average or the whole month.

But if you-- total aside, sorry, but this guy's site is a cool way to look at the fees. So you can see here's-- he sort of organizes transactions by fee rate. It's too low-res to really get everything, but if you just search for Johoe-- J-O-H-O-E-- he works on Bitcoin stuff and he made this cool site, which is open source and you could even run it on your own node if you wanted to, and generate the same cool JavaScript color-y things and you can see the fee market.

And I'm not an economist, but it is really interesting seeing there's clearly market failures occurring here in that-- so, you can pay the green-- you can pay 10 Satoshis per byte, and you'll get confirmed in 10 minutes. Or you can pay 1,000 Satoshis per byte, and you will also get confirmed in 10 minutes. And most people are paying 10, but someone's paying 1,000. You know, it's got the whole spectrum. You've got multiple orders of magnitude of people paying for the exact same thing, and they can all see each other. It's just a weird sort of-- seems broken.

And part of it is just the cost to write the software. If you're an exchange and everyone's sending you support requests, and this happens-- OK, I don't know just, pay a 500 Satoshi per byte fee-- and then it seems to work. And, yeah, we're losing a couple of thousand bucks a day, but let's just not deal with that.

And I think that's part of it, is that there's a lot of software out there that just has a fixed fee rate, or even a fixed fee, regardless of how big the transaction is. There's a lot of software that, years ago, wouldn't have to deal with this issue because there wasn't really competition to get into a block, and now they do. So it's kind of cool to look at and see the history of it. But

I'll get into depth of fees and stuff in, I think, two weeks or something.

OK, any questions about-- yeah.

AUDIENCE: What was that website again?

TADGE DRYJA: Well, it's in some Dutch, or something. Just search J-O-H-O-E, Johoe. He's the guy. It's the first thing on Google. J-O-H-O-E is his Dutch nickname, or something, I don't know. He's a cool guy.

He actually-- I think I talked-- did I talk-- there was some randomness problems at some site. He stole a bunch of Bitcoins and gave them back to their owners. He found there was a K reuse, like nonce reuse vulnerability in some wallets. And so he's like, hey, look, there's like 100 Bitcoins that I can just take because I can calculate the private key. And he took them, and he was like, I think I know who these are and can you prove it, and then I'll give them back?

So he sort of grabbed-- you know, finding a wallet with, like, thousands of dollars coming out of it on the street, he grabbed them all and tried to get them back to people. I don't know the guy. I've never met him but seems like a nice guy. Anyway.

[LAUGHTER]

That's how Bitcoin works. You don't meet anyone, but you see these people-- oh he's a nice guy. Oh, he's a jerk. The weekend was kind of interesting over Twitter, but anyway--

AUDIENCE: I saw that.

TADGE DRYJA: Yeah. OK, so you build these transactions. There are issues here. Two inputs, two outputs-- that's going to be kind of big. You're going to have two different signatures. It's going to be a little bit higher fee. What would work better than this? It's kind of a silly question. What would work better than having two inputs and two outputs to make this transaction to pay someone six coins? Yeah?

AUDIENCE: Maybe if you wanted to have an anonymous transaction doing something like multiple transactions in smaller sizes?

TADGE DRYJA: Sure. Yeah, that's-- you could send-- so, that's actually two slides from now. The next slide was just, well, what if you had a UTXO that was exactly the right size? Then it's easy. You just

send them the six coins.

If you have the exact right size UTXO in your wallet, great. You just send it over. It's like if you go to a shop and they're like, OK, that's $10 for lunch. You're like, great, I have a $10 bill. Here it is. We don't need to deal with pennies and quarters and stuff. It's annoying. So sometimes this happens. It's great. Generally, it won't. Generally, you will have change and multiple inputs and outputs and it's kind of annoying.

So coin selection is a tricky problem. For CSE terms it's NP-hard, actually, but there's heuristics that work OK. If you have a ton of UTXOs and you have to send these payments, you can actually, in a reasonable amount of time, calculate the optimal way to do it. But there's some heuristics at work, and the question is what are we optimizing for?

Generally, you want to optimize-- minimize the number of inputs used. The inputs are much bigger, they're going to be like a hundred something bytes, and the outputs are pretty small, they're like 20-30 bytes. So if you want to minimize size of your transaction, minimize the number of inputs, which is easy. You just pick your biggest UTXO and spend that one.

Yes?

AUDIENCE: Isn't it like the knapsack problem, though?

TADGE DRYJA: Yeah, it basically is, yeah. Well, because it's multi-iteration, if you're just trying to optimize your transaction right now, you just use your biggest UTXO.

So for example, it's sort of the analogy of you're at a checkout counter and someone says, OK, that's $12.93. If you want to minimize the number of bills you're handing to the cashier, you just take the 100 out of your wallet. That'll always work. You just say, I take my biggest bill, hand it to you, OK, I'm minimizing the amount of bills I'm handing you in this one transaction.

However, that could result in a whole bunch of bills coming back, a bunch of weird change. And then also, long-term that doesn't work. If your strategy is always just hand over the 100, or you go through your wallet and just hand over the biggest bill you have every time, no matter what they ask, that's super suboptimal because if they say $12.93, and you have a 20 and 100 and you hand over the 100 like, why did you do that? And then you're going to have four 20s.

So, it's very similar to that, except now that the change and bills have arbitrary denominations.

There isn't a fixed-- you have 100s, 50s, 20s, 10s, 5s. Now it can be any number.

So if you're just looking at one time, just pick your biggest UTXO, you'll have the smallest transaction.

But you want to minimize next time, so you ideally can eliminate the change output and get you a perfect target. It's actually really complicated. There's really cool research on how do we select coins for long term? Yeah?

AUDIENCE: So why don't you just take the biggest UTXO that's larger-- or, the smallest UTXO that's larger than your output size?

TADGE DRYJA: Yep, that can work. That's not-- that's a good heuristic. That's a good-- pretty easy to code, sort your UTXOs, go here, use that one.

It's not really optimal because then-- it's a lot better than taking big ones-- what do I have in my wallet? So I've actually written an SPV wallet and all this stuff just from scratch, and it's kind of interesting. You learn a lot about how it works.

I target two inputs instead of one, because then eventually-- if you do that, what will happen is you're going to be using one input, which is great, and then you're going to run out of big inputs. And then you're going to always have to use two or three, and you can get a lot of dust. Dust is like the colloquial term for really small UTXOs, where you've got a bunch of pennies. So that's one issue.

Another issue is privacy concerns. When you use two UTXOs or have two inputs in the same transaction, that's linking those transactions, linking those two UTXOs. It's not definitive. You can interactively create transactions with other people. In practice, that doesn't happen.

You could say, hey, I want to pay Alice five coins, and you want to pay Bob six coins, and let's put my two UTXOs and your two UTXOs and we'll pay these two people, and we'll put our own change outputs, and we'll sort of mix this transaction together and we'll all sign it. And you can do that securely since you only sign when it all looks right to you, and everyone only signs when it's done.

But the coordination problem is pretty severe. You have to find other people who want to make transactions at the same time that you do. It's annoying. So, in practice, since you're just using your wallet, if you see a transaction with multiple inputs, you can you can surmise, OK, those

are the same person or the same company.

And if you want privacy, if you want maximum anonymity what kind of coin selection or payment strategy would you use?

AUDIENCE: Would you make just a bunch of transactions?

TADGE DRYJA: Yeah. If someone says, hey, pay me six coins, well I have these three inputs, and I'm paying you two coins here and one coin here, and three coins here. And I paid you six coins but in three completely separate transactions.

That no one does either because it's annoying. You could. It would be the most anonymous, but even then, what if they all happened at the same time, and you see they all get in the same block? And you're like, OK, well they're not linked nearly as closely, but I am seeing that these three transactions happened temporally similar times. So there's all sorts of things to try to optimize for.

OK, any other questions about-- yeah?

AUDIENCE: So does this mean that every time people are going to have a smaller and smaller split of a Bitcoin in their wallets? They're just going to have smaller and smaller amounts because you're going to-- if you have a $100 bill and then you're paying $20, then you're going to get four other $20 bills.

TADGE DRYJA: Yeah.

AUDIENCE: Eventually you're just going to have smaller and smaller-- is that a fair implication, or--

TADGE DRYJA: OK, short-term, yes. If you start out with a bunch of coins then start using it, yes. But it does reach equilibrium in that let's say you've got all these little tiny outputs-- you've got all these $1 bills-- but then you need to buy something that is 20 bucks. You have 20 inputs and one output. And whoever you're sending to now gets that one big output.

And so, yeah, if you graph that over time, initially everyone was getting-- all their outputs were 50 coins each because that's how you mined them. And now they're getting smaller, but there is sort of an equilibrium after you've used it for a while. Any other questions about sort of coin selection, UTXO selection? Yes.

AUDIENCE: According to a news report, I guess if you [INAUDIBLE] transaction, you also have

[INAUDIBLE] transactions.

TADGE DRYJA: Yeah, yeah. So that's costly. That's another reason probably people don't do this. I think the biggest reason is it's just annoying to code, and you can have failures where like-- here, give me six coins. OK, I'll give you 3, 2, and 1. Oh, the 3 didn't work, but the 2 and 1 did. Well, now what do we do? You paid me half of it. It's nice to have all or nothing payments. And also we have to send different addresses. There's all sorts of things. Also, it will be higher fees. In practice, it's actually not much higher.

Let's say having three one input, one output transactions versus one three input, three output transaction-- you don't save too much space. Most of the space is taken by the inputs. And the overhead for a transaction is only 10 or 20 bytes or something, so it is not a huge difference.

The main difference is that you're never coalescing into larger output sizes, so you're going to always have to sign since you going to have more inputs overall. This is a really, kind of, cool problem. There's a lot of computer science-y stuff but a lot of heuristics and how people use it.

Also, the fact that fees are variable over time means you might want different strategies when fees are low versus when fees are high. So when fees are low now, I should make-- or maybe I just make a transaction to myself where I condense all my little $2 outputs into one big $1,000 output so that when, later on, if fees are higher, I want to spend my money, I can do so more efficiently.

And there is evidence of this with exchanges and stuff where a lot of times fees will be lower on the weekends because people aren't buying and spending Bitcoin as much, I guess. And so certain companies would say, OK, over the weekends we're going to sort of condense all our UTXOs and combine them and then we can make smaller transactions during the week.

So there's all sorts of cool strategies here. It's an interesting topic. I haven't gone super in-depth, but the guys that chain code work on it. There's a lot of discussion about it, so it's kind of cool.

OK, I'm a little bit behind. OK, next we'll talk about losing money, and that's another really important part of detecting the blockchain. It's hard to do, but you have to detect when you've lost money. And it's tricky because just because you signed the transaction doesn't really mean your money is gone. You can't just unilaterally say, OK, well, I'm making this. I signed it. There, my money is gone from my wallet.

Well, not necessarily. Maybe this never gets confirmed. So maybe you still have that money. So you broadcast it, but you sort of have to wait until it gets into a block, and you also need to listen for your own UTXOs, even if you haven't made a transaction, and see if they've gotten spent. And why would that be? Can anyone think of a reason why? I haven't signed anything, as far as my program is concerned, but I might lose money anyway. Why would that be? Yeah.

AUDIENCE: Well, one reason is you get hacked.

TADGE DRYJA: Sure, you get hacked. That's the bad reason. A good reason is, well maybe you have the same wallet on multiple computers. You've got the same keys. So, getting hacked is sort of a malicious instance of this problem where I thought the wallet was only on my computer, but actually, someone else has a copy.

But even non-maliciously, I've got a copy on my phone and I've got a copy on my computer. It's the same addresses, the same keys, the same UTXOs. That's totally doable. And then when I spend money with my phone and get to my desktop, my desktop needs to sort of download and see oh, money got-- you know, you lost money. And it's like, oh, yeah, yeah, I remember spending that. So you can have that over multiple computers.

So if you're designing wallet software, you do definitely need to make sure that even if it's unexpected from the wallet itself, and it doesn't seem like I generated a transaction, there can still be a transaction taking your money away.

Wallets without Bitcoin, and that's sort of a cheeky phrase. OK, I don't mean they don't have any Bitcoins in their wallets, I mean they're not running Bitcoin in the same sense that we've talked of.

So we talked about running Bitcoin where you download the software, you get the headers, you verify all the signatures, you build the UTXO set. Can you use Bitcoin without doing this? What do you guys think? What's a simple way to possibly use Bitcoin without having to do all these things? So, a really, really simple way? If you don't want to do work, what's the simplest way to not have to do work? Get someone else to do it, right.

So, for example, my dad has Bitcoin, but he just gives it-- he's like, you deal with it. So I've got a couple of Bitcoins that's my dad's, and I have to make sure like, no, this is not my money. Yeah, get someone else to do it, right? So that's what we're going to talk about, the different

ways to get someone else to do this.

And what we called before, running Bitcoin, many now call a "full node." And there's also the idea of a "light node" or "SPV node," which we'll talk about.

Some people don't really like this distinction, and it's like, well, wait. Full node is running Bitcoin. These other things, we shouldn't have to call it a full node. We should just call this a Bitcoin node and these other things are not quite there.

I will prefix there's a lot of argument about terms in this space. So there's some people who say, SPV doesn't exist. And other people, this isn't SPV. So people argue about the words. It's not like we have really nice, definitive terms. I'm generally trying to use the most widely used terms, but there's probably people who will take issue with it, so sorry.

So, SPV is sort of a step down below running a full node in terms of security. It's called Simplified Payment Verification. It's written up in the white paper on how to do it. And you can verify all the work without doing too much signature verification or having too much data.

So the basic idea is you're optimizing for not having to download as much and not having to store as much at the cost of some security, and I'll talk about those costs.

OK, so before we have this list of what you do for a full node, the SPV method is a bit different. You still do the same part in the beginning. You connect, you get your headers, you verify all the work. OK, cool.

The next step, you tell another node that you're connected to all of your addresses, all of the public keys that you've ever generated. You tell it to them.

Then, for each header you go through-- and instead of downloading the whole block and getting all the transactions and verifying them, you ask the other node, hey, did I get any money, or did I lose any money in this block because I've told you all my addresses? Oh, sorry. You also tell them all your UTXOs.

You also tell him, here's all the money I have, here's all the addresses I could possibly receive money on, did I get or lose any money in this block? And then they will return to you a Merkle proof of the transactions where they think, yeah, you got some money here, or yeah, you lost some money here, and you can verify this. Yes?

AUDIENCE: What's the other nodes' incentive to respond to you?

TADGE DRYJA: There is none. You're not paying them. They don't know who you are. There's sort of a meta incentive in that I run a node that will provide these Merkle proofs because it's like, well, it helps Bitcoin, and maybe if I have some Bitcoin and I'm helping other people use it, my Bitcoin will be worth more.

But that's a pretty diffuse sort of thing. And it can be problematic because some of these things-- I didn't mention that in these slides, but the server side can get a little bit costly in terms of CPU because you're potentially-- as a server-- the client requests hey, here's this block. Can you filter it for me, find things that I'm looking for.

So now you have to load that block into memory, look through it. It's not too CPU intensive, but it can be-- you know, when you have a bunch of them, like 20 or 30 of them connecting to you-- I've gotten 30%, 40% CPU for doing this kind of thing to serve other users.

Most-- almost all phone wallets-- well, many phone wallets and many desktop wallets are using this model, and so you'll see-- for example, so here's a full node in this building. I actually rebooted it recently, so there's not very many connections incoming.

In practice, these two are actual full nodes, I bet. This is a fake node. This is a fake node. This is-- they're all fake, yeah. Well, sorry-- these are all-- no, that one may be not. Well, you can look. But a lot of nodes will say they're nodes and they're not, and they're just trying to track where transactions are coming from and keep track tabs on you and stuff.

And these are SPV nodes, these bitcore, because they don't really ask for-- I don't know what they're doing. They're not asking for anything. So you can look through all the messages. I think Ethan will talk about this a bit more Wednesday, but there are a lot of SPV nodes. There's a lot of stuff out on the network and you have no idea what it's doing, but it's pretty clearly not running a Bitcoin node.

So, yeah so I'll go through these steps a little bit. Oh, yeah. So the Merkle verification we talked about last week, where, if there's a block and there's thousands of transactions in it and this server wants to prove that one of these transactions is yours and is in there, you say, OK, here's my transaction. They just need to provide you this transaction ID, this hash, and then you're able to see, OK, yeah, it was in the header. So my transaction is in there, you're not just making it up.

I didn't talk about the good part. Well, the good part is you don't really need to maintain a UTXO set and it's pretty small, so it saves space, saves time.

What are the problems? There's a lot, and I definitely admit before writing my own SPV wallet code, I didn't think there were a lot of problems with it. I thought it was like, oh this is SPV, this is cool. This is how wallets work. But when writing the code myself, I'm like wait, this is horrible. What do we do?

OK, so the first thing you do is you connect, you get the headers, you verify them. This is exactly the same procedure as what a full node does so there's no difference, it works. No difference there.

The next step, you tell a node all of your addresses. What? There goes all your privacy, right, because you're just connecting to a computer. You have no idea who they are, who's running it, and you're telling them hey, here's all of my addresses, and also here's how much money I have. Here's all my UTXOs.

You can lie. You can add things that are not-- you can also add some addresses that aren't yours, or add some UTXOs that aren't yours, and you'll get some transactions back that you can then filter out on your own. So you can you can raise the rate of false positives for that server.

And so there's these Bloom filters that are in the Bitcoin Core code. They said the idea was well, you can sort of dial your own false positive rate. I'm not going to go into Bloom filters work. If you've used those in other classes, cool. But it basically gives some data which allows people to match things. But they don't in practice have good privacy.

You can create a Bloom filter where they've got 10% false positive rate. And so when the server says, oh, looks like their transaction, maybe it's not because 10% of the time it's just a false positive. However, when you have really high false positives, you lose all the efficiency savings of SPV and it sort of cascades where you've got these false positives and the server thinks, oh, you got money, but it's a false positive. And they add that "you got money" into the Bloom filter itself and the Bloom filter can really quickly become saturated, and then they just start giving you everything.

So in practice, and there's some papers about how the people who put the Bloom filters into Bitcoin thought, oh this is good for privacy, it's fine, and in practice, it really is not good for

privacy. So you end up basically telling a node all your addresses.

And there's research on how to do this in a better way, and it's one of those kind of things where some random anonymous person with a, I think, inappropriate swear word email address posted to the mailing list and said, hey why don't you guys do it this way? And it was like, oh, yeah, we should have done it that way, oops.

The basic idea is instead of creating a Bloom filter as a client sending it to a server, basically instead of telling the node all your addresses and asking, what the nodes will do-- the full nodes-- will create a Bloom filter based on the entire block. And then the client can retrieve that, match that against their addresses, and see, hey, did this block have anything of interest to me? And if so, request it-- much better privacy at a pretty small cost in overhead. And so, just no one thought of it. There's a lot of things in Bitcoin where it's like, no one thought of it, we did something dumb. And then something better came out and now we're working on it.

OK, so you tell the node all your addresses. That's a problem. For each header, ask if you gained or lost you UTXOs? So can you think of any problems here? Yeah.

AUDIENCE: Could they lie and not pay some of them?

TADGE DRYJA: Yup. Easy to lie. You just don't tell them. If you're a server, you just omit things, and you can maybe mitigate that by connecting to a bunch of different nodes but then you lose even more privacy because you've now shared all your addresses and money with multiple anonymous nodes.

But it's really easy to lie by omission. Someone says, hey, here's all my addresses, OK, did I get any money? Yup, yup. And then you see one where they got a bunch of money and just don't tell them. And they don't know.

This can be annoying in regular wallets in the Lightning Network stuff that I work on that I'll talk about, hopefully, later. This can actually be very damaging. You can lose money because of this. But, in general, in Bitcoin, you won't lose money because you're not aware of a transaction.

So this is also a problem, easy to lie by omission. The Merkle proofs help, but they prove inclusion, not exclusion. There's no way to construct a proof that-- I'm going to I'm going to give you proof that I'm not omitting anything. Although, with the idea of the block-based filters

sending, there are ways to construct that, so it's even better in that sense. OK, so these are some of the disadvantages of SPV. Can anyone think of any other problems with it, or-- yeah?

AUDIENCE: Fee estimation.

TADGE DRYJA: Yeah, OK. So, yeah, you don't know-- since you're not downloading the blocks, you don't really know how much fees other people are paying. You're not verifying. So even when you get transactions, you cannot verify any signatures because you don't have UTXO sets, so you just see that it came from somewhere, but you don't know if the thing it's spending even exists or has a key or anything, so you can't verify the signature.

You don't know how much money was coming in, so even if you look at the transactions, you can't tell what fees they're paying. You sort of can if you download the entire block. There's ways around it, but it's really ugly, so it can be very difficult to estimate fees. So, in practice, you'd probably ask the same server that you've told all your addresses and all your UTXOs to, hey, what fee should I use, then they tell you that. The idea is, well, if I ask five people, hopefully, most of them will be around the same. So there's a bunch of problems with SPV.

OK, so SPV sounds pretty bad, right? I think I'll stick to my full node. But is there anything worse than SPV? Asking for a friend. Can I go worse? So does anyone know something we can do that's worse security, worse privacy than SPV and that's also very popular? Yeah.

AUDIENCE: [INAUDIBLE]

TADGE DRYJA: Yeah, that's even worse. But, yeah there's a step in between. So you can take out some of these steps where you just use an API and you just ask people. You have a website, blockchain.info or Mycelium Wallet, or bunch of wallets-- BitPay's, Copay, things like that where you don't verify any headers, you don't look at any Merkle proofs, you just skip right to the tell the remote node all your addresses and UTXOs and ask how much money you've gained or lost.

So you've sort of outsourced the entire process. You don't store really anything on your computer. And you say, well, but you do have your private keys. You say, I made some private keys, I made some addresses, and then I tell this website, hey, here's all my addresses, how much money do I have? And the servers responds, yeah, you've got UTXOs, cool. So then you can build the transaction, sign them, and send them to the server.

So what are some advantages and disadvantages of this? There's probably some obvious

disadvantages, right? Can anyone think of an attack that this does not help you against? Yeah.

AUDIENCE: You can just make up transactions.

TADGE DRYJA: The server can just say, hey, you've got 1,000 Bitcoins. You're like, awesome, but it's just completely made up. As the client, you don't verify anything about these transactions. So that's a pretty big problem.

And the thing is, in practice, one of the issues is that people are generally not as aware of these types of attacks because mostly people worry about spending their money, and they don't really-- merchants worry about charge-backs and worry about receiving and verifying that they've received funds all the time, but most people's experience is they get paid once a month or twice a month with a paycheck, and the money shows up in their bank or whatever, and they never really worry about that. They worry about spending their money and getting defrauded or things like that. So it's not something a lot of people think about all the time is, did I actually get paid?

So there's easy fraud that you can do with this kind of attack vector where you sell a car on Craigslist, and someone comes and says, yeah, I paid you the Bitcoins, but they've actually compromised the server and you haven't gotten paid at all. But you think you have, so you give over the goods. So, yeah potential problems-- they can say you got paid when you didn't, they can say you lost money when you didn't.

And if it's in a browser, that's even more fun because they can change the code. The JavaScript is not pinned to anything, so if someone compromises that server, they can change the code and potentially get your private keys. So you have, really, very little security. The blockchain is not really providing anything in this case.

However, this is much more popular than running a full or SPV node, because you know, blockchain.info, you just sign in, there's a lot of wallets on the phones that work this way as well. And you do at least have your private keys, hopefully. So you've got that, right? You're not giving custody in any sense to them but they learn a lot of information.

OK, so not even SPV. Can we do worse? Yeah, so the Coinbase company was an example of "can we do worse?" Yes, you can. Someone else's coins is worse. The case where my dad said, hey can you hold on to these coins for me, it's worse. He doesn't run a node, he doesn't

have his private keys, he doesn't really understand Bitcoin that well. He wants to, but he's busy and he's like, hey, you know this stuff, you deal with it. You know way more about this than I do. I trust you since, you know, we're dad and son and stuff so not a huge trust problem there, so I do it for him.

But you know, banks, right? So the idea of a site or an exchange or something like this where you don't even have your private keys. You just have a website where they run a node and a wallet and they owe you the money.

It tends to end badly, and even if it doesn't end badly, it misses the point. The whole idea of Bitcoin was like, hey, you can have your own money. It's kind of cool. It's running on your computer. It feels like it's missing the point to just hand it over to some bank. And it's not even a bank.

Most of these sites, a big reason why it tends to end badly is there aren't the same protections. Banks have to do a lot of work, and there's FDIC, there's all sorts of rules, and they also build these big structures with really heavy stone pillars, so you're like, yeah, they can't run off because this bank's not going to move. It's made out of rocks. And the banks in Bitcoin do not have big stone pillars. IP addresses are really easy to change and move the computers around.

Another thing, they're running a node, right, these Bitcoin banks that hold all your funds. Sometimes they don't, so these banks themselves might run SPV nodes or API things. I don't want to name any names, but there's pretty good evidence that big exchanges might even just connect to an API and not even run their own node.

Another-- there's a lot of things like this where, when something bad doesn't happen, people just keep pushing it-- where miners themselves don't verify the blocks because they think, well, he must have created a valid block and I'm not going to verify it and everything works.

So the other thing is, while it sounds really bad, in practice, there haven't been really many SPV attacks or API attacks. We know of this, but in practice it's hard to do. If you want to defraud someone by compromising blockchain.info, you have to compromise blockchain.info. You don't have to do all the proof of work, because they're not validating it, but it's still hard to do and it requires a coordinated active attacker with quite a bit of resources.

And so when it doesn't happen, people say, well, SPV is just as good. We don't have any

evidence of people being defrauded, so it's just as good. But that is kind of dangerous because when everyone starts doing it, you start to lose these protections. Any questions about the someone else's coins model? There's all sorts of legal issues. There's a very long list of ways it ends badly. I don't know-- what is the half life of a custodial exchange in Bitcoin? It's like a year or two, and they drop off.

So why do people do this? And here's a table of trade-offs with these things. It's mainly convenience, and so that's a real reason to do it. So if you're running a full node, you're going to have to download at least 170 gigabytes. That's a lot, right? It's going to take a while.

Storage-- you're going to have to store at least 4 gigabytes long term. And that's going up, but not going up too much. It's actually gone down the last few weeks. That's the UTXO set you have to keep track of. You don't have to keep track of this 170 gigabytes. It, by default, does but you can turn on pruning. But that's also super user-unfriendly. You have to edit a bitcoin.conf file and type "pruning=500" or something, and then save it, and then it'll prune down to 4 gigs. There's no-- at least, that I'm aware of-- there's no GUI nice menu thing where you can say, hey, I want to enable pruning. I don't have to store it.

Speed-- on a really nice computer, it will take at least six hours to download all this and verify it. That's pretty impressive because it used to be more, but that's still six hours. People don't want to deal with that.

Privacy-- certainly, we can do better. There's a lot of research on how to make privacy and Bitcoin better but this is what we got.

Security-- this is as good as we've got

So then you go down to SPV. Network-- you only have to download about 50 megs, all those headers. If you've got a wallet with lots of transactions, you're going to download 100, 200, 300 megs because you're going to have to download all the transactions that pay you or you're paying out to.

Speed-- I said seconds, I think I want to change it to minutes. It's not that fast. It's a lot faster. I think seconds is an exaggeration. Well, it's like 60 seconds. Anyway, it's pretty fast. You download all the headers. That takes the same amount of time, but that can be a minute or two, and then you're syncing the blocks. It's really quick, they're small.

Privacy is poor. You lose a lot of your privacy in SPV because you're basically telling random

computers on the internet, hey, here's all my money. Hey, here's all my addresses. You're not completely losing everything, but it's pretty easy for actors to reconstruct your wallet from that.

Security-- medium. I don't know, there haven't been any real attacks on this, but you're not verifying the rules of Bitcoin. If everyone's running SPV, then a miner can say, hey, I just generated 1,000 coins out of nowhere, and no one's looking for that transaction. It only pays me, and no one's going to see that and reject the block. So if everyone runs SPV, you're not checking up on the miners, which is a very real threat. Miners do crazy stuff and you got to watch out for them. So, security-- questionable.

API query, where you just ask a website hey, here's all my addresses, how much money do I have?

Network traffic-- I don't know, less than a megabyte. You have to load the websites and stuff but it's pretty light.

Storage-- you basically don't have to store anything. I mean, you have to store your private keys, but those can be password-based and derived on the fly.

Speed-- like a second, right. It's real quick. You're making an HTTP query, you're getting a response, you're parsing it, it's real quick.

Privacy is poor. It's worse than SPV, but because it's really easy because you just hand them over all your addresses in the clear.

And security is also quite poor, in that they can say hey, you got money or you lost money, and you just accept what they say.

Hold my key-- this is network traffic. I don't know, you have to go to a website, I guess. There's no storage, there's no speed, there's no privacy, there's no security. You're just handing the entire thing off to someone else.

So what would you guys guess are the popularity of the different models? Most popular to least popular. Yeah, this is definitely the most popular. Second most, third most, fourth most. Everyone does this, a few people do this, some people do this, and a couple thousand people do this.

It's a problem, and this is something that is one of the ongoing problems, not just in Bitcoin.

Ethereum would be a little different, but still, it's going to be a lot of this. Ethereum has a weird different SPV. There's other models. There's one in the middle for Ethereum that's also quite popular. It's like SPV with UTXO commitment. Well, no, I guess it would be more here. Anyway, I'm not going to go into Ethereum.

But it's a problem and there's different ways to attack it. One of the issues is that a lot of people who program Bitcoin itself really only focus on this, and they say, look, this is not our problem. We can't solve this.

We're going to try to make this-- the way we're going to try to solve this, is let's try to get the speed down. If it takes days, people are going to move this way. If it takes hours, maybe a lot of people will say, hey I was using SPV, but yeah, it's not too bad running this. I'm going to run this and get the more security. Let's try to keep this number down. Let's try to keep speed down. Let's try to improve privacy and security of the full node.

That is generally what most of the Bitcoin Core developers focus on, which I don't argue with. But it does lead to some neglect of SPV, where there's not-- it's been over a year, year and a half, almost two years where we know how to make SPV better and more secure, but there's not a lot of enthusiasm and people working on it. And people argue about the security of these things.

This, there's not much you can do. I mean, there is kind of cryptography research like, hey, is there some cool way I can send you all my addresses so that you can figure out how much money I have without you learning all my addresses? That's called private information retrieval, and there's all sorts of papers on that. In practice, there aren't any that use that.

And this, well, yeah multi-sig. That's more like regulation. Can we have restrictions and rules on these, essentially, banks to try to make it safer, maybe? These are the two where software development can definitely help make this a lot easier to use.

So we can encourage people to use it, but most people-- security is hard because most people, if you don't see a problem, this is a lot easier, and a lot of people think, well, I'm not good at computers, so-- they are, and it's safer if I give all my money to someone else. In some cases, it could be true.

But that has systemic effects where now you've got these five computers in the world, and if you're able to compromise those, they have just billions of dollars worth of Bitcoins on them.

And so that's why black hat hacker kind of people, it's like the best thing to do. It's like, there's a computer somewhere and it's got a billion dollars of untraceable money that I can just steal. Like, it's-- what could be better? Yeah, I could get everyone's passwords. That's cool. Or yeah, I could read people's emails. Whatever. Or, I can just steal a billion dollars. So what do you think they're going to do? So this leads to huge concentrations of coins in a very small number of nodes, and people try to attack it.

So this is sort of the landscape we're in now. It's certainly not ideal. There's a lot of technology that's pretty good that's not being used. There's a lot of technology that's crummy that's being used a lot and how we make this stronger and faster, how to make this faster, things like that are really interesting research areas.

Almost done. Wallets are fun, but usability issues. If you want to try testing out wallets you can try downloading them, playing around with them. They often leave quite a bit to be desired. The one I work on, Lit, leaves enormous amounts to be desired. It's all in text.

And Ethan should be here Wednesday and good luck with the problem set.