### Security\_week5\_Tutorial5\_1-20241008

说话人1 00:00  
Before we hit the lecture five, I just want to go back a little bit to the number three readings is to address some general questions. So first of just at the end of last week's lecture, if you haven't found it in, there are some additional examples. And these examples match to the two longer examples for rsa an album or so. They like full workout examples. There's two for calculating the modular index, these two for doing the square and multiply the algorithm. I think the one is, I think this one is the one for the longer rsa example, and then the album on one is the longer, slightly more difficult one. So I think the example that was in the slide, the final number you end up with here is a negative.

And then you cannot use a negative number, because in practice, it might be congruent. But if we use it for rsi then it's gonna be an exponent. We don't want a negative exponent. We're always looking for the smaller positive numbers, even though our modular inverse might be negative after we do the extended liquidity algorithm. Remember, to always change it to the smallest congruent positive number. You can do that just by adding the modulus to it. If it's a negative number, you can add the modulus. And then you can get until you get the school is a positive number. So that was the one question. While i'm at this point, as we are using numbers and we're getting closer to that term, if there is a 2 weeks away, you can use calculate there in the winter. As long as it's like a normal calculator, like when you had in high school, the one that you can add and multiply and do things with the conscious, it just cannot be like a very, a fancy dropping calculator, the ones that you've been programmed and put notes into the things like that.

As long as it's a normal calculator, that is fine.

Then the second one, I just want to do one example of the square multiply, because the example we did in class, 11 to the power 15. That one, just so happened to work out that way that we actually needed all the terms. So it was sort of 8 + 4 + 2 + 1.

But this does not always happen. So it might not have been entirely clear what happens with the terms and how we collect terms together. If we have the exponent, remember, from the square multiply algorithm, we try to take the exponent. We reflected a as a sum of power of two. If the exponent is very small, this would be quite easy. But if the exponent becomes a little bit more large, it might be easier just to convert the decimal exposure into a binary value.

That will give you the different exponents you want is the first one would be a one and two or four, a 60, 3024, and a 158. Right?

You can see that very clearly that 130 is 128 + 2. You still have to do all the squares, though, I have 17 to the power, 130. Is it 17 to the 128? I'm 17 squared. Then you will do all the square terms. 17 squared is equal to 289. Modular 11 is three, and then the 17 to the 4 is 17 squared, all squared that you already calculated the 17 squared. You can speak your previous answer, and you square that. You get nine. 17 to the 8th is just 17 to the four, all squared. You can use your previous nine. I squared, 1 ~ 0, 11 is 4. We keep doing the same thing. I thought we get to the highest term that we need is 70 to the power 128. Right? So now you have and sort of what are the reductions for all your squares? 17 squared, 17 to the 4 to the 8th to the 16, 32, 64, and monthly average. 945394.

Then in the last step, you only need 17, 28, and 17 squared. You actually only need this one, this one, the other ones you use, because you needed to calculate 1752 ~ 28. But you don't really need it. You're only going to use equation 17, and 17 to the 130 modular 11 and 7, 24 × 3, modular 11, which is just equal 2 points.

I'm just, in case that was a deal on how we go back. So we always do all the squares, but we might not need all the terms when we do the final calculation, ok so that's what I want to say.

For today, we're doing some data integrity. Before we get here, please remember your problem set one. It is due next week, 2 o'clock. You have to upload it on canvas. It's already been on canvas, I think, for the last 2 or 3 weeks.

At 3 weeks, I think. So you can get the problem set from there. It has quite big numbers for its problem for its number theory equations.

And as I promised you before, after problem set one, day one, dp numbers and number theory problems, and there won't be specifically number theory problems anymore. Either just to force you to try and study that and practice it for the tutorial arrangement. Remember, when we had to make up class on sunday, I said we can only have one tutorial. I thought it worked quite well. There were actually many more people than final money here in total. I want to keep this arrangement for now unless somebody lets me know that they don't like it. Okay? So I think for today, we can also try having only the one tutorial session. We'll have one from 4 to 5 after the lecture ends, and we'll just stay in this classroom. Because otherwise, I think for the guys who are normally in the second tutorial, they said more of them come down. And everybody that was coming to, the first one, anyone so comes. So it's dried up and it's a bit easier for everyone. No one has to be disappeared for an hour and they come back.

So 1 to 4, if you dont like it, you can get enough im open to to suggestions, but this seems to be the one that gets the most even competitive politician. So let's try this for a week or two. See our best. For the previous lecture, we looked at asymmetric encryption. We looked at the difference between symmetric and asymmetric models. Remember, when we looked at symmetric encryption, we had alice involved. It was a symmetric system, because alice and bob had the same key. Alice, we've made message with the key. Rob would use the same key to recover. The plane takes from that side of this message.

In the asymmetric model, there are also two keys, but they are not the same. They also don't belong to two people. Right? Bob will have two keys, a public and a private key. Bob will keep his private key, a secret. You will not tell anyone, and you will take the public key and you can send it to anyone he wants in any way he wants. Right? It doesn't matter if other people see it doesn't make it insecure like that's how it's supposed to work. So anyone that has the public key can encrypt the message for bob, but only bob can decrypt. So any one of the public key can encrypt. Only the person with the private key can decrypt.

Okay, we then said the unlike the symmetric cycle, as we look at, it was a lot of the label winery shuffling, permutation, x or rotation, all sorts of things. It's a major encryption, pretty much was number theory.

We could write everything as need to do equations, and they were based on what we would call one way mathematical functions. We look at two of these. We look at the factorization problem. If I would give you two prime numbers, very easy to multiply them together. If I gave me the product of the two primes, very difficult to factorize, right? And then that's the case. The numbers are adequately large. Right? We also look at the discrete logarithm problem. We said, if you have the base a the exponent b and the modulus in calculating a to the power of the more aid is very easy. But if I gave you the result and the base and the modulus, calculating the exponent that gave us that result would be very hard. That was the discrete operative problem. We ended the three algorithms, rsa which was used in that nature for data encryption. And the security is based on the factorization problem.

Then we had with this quotient, once you live in business and extended to the algorithm, that's basically the skills you need to work with rsa not even properly distortion. You just have to remember that phi of n is p minus one two, minus 1. You don't even have to remember like the p with re is equal to p to the u minus one times p minus one is that p minus one times q minus one. And then the margin that are in this, of course, one to the reduction as well. And then for other mall also used in that version that we look at what data encryption it is related. Discrete logarithm problem. And for doing that, we only need the modular inverse. And the modular is appreciation.

And then finally, the development, the phenomenon was a bit special because it couldn't really be used to encrypt anything. But it could be used to exchange two symmetric shared keys between alice and bob. So it's very much a key management or a key exchange algorithm for me. Right? It was based on something called the different common problem, but we say that it is basically equivalent to the discrete number of the problem. Right? Because you had sort of e to the power, a mod, e and g to the pb mod p and then the key would be t to the ab going from g to the a and g to the bg to the ab is the common problem.

But essentially, the easiest way to solve it is to find other a and b right? And that's the exponent. And that's protected by the discrete talk with the problem, be very hard. And then we look at men in the middle of attack and we say, anyway, to solve that is to put some data origin of education into the protocol, provide. That's okay.

So again, for the number theory, after the problem set onwards, you need to do modular exponentiation. You need that rsl of them all behold, then you need to be able to calculate the modular investors. If you need that rsn on the mall, you need to do for this potion. But is this just for rsa as I said, it's just p minus one plus q minus one. Don't focus on the number theory or proving number theory things. I'm just focused on being able to do rsa problem on the more problem, just being on the problem. That's where the focus should be. In today's lecture. Are we going to look at data integrity in our creation? We're going to look at some digital signatures, some past functions, and some messages, indication codes. This cover is silent, two in silent five technology that impacts systems and security mechanisms.

Right up to now, we've pretty much been still speaking about confidentiality only. So integrity is not the next security services we can want to. So what does integrity provide for us? Essentially, it prevents any type of modification of data. Somebody that leads part of the data is somebody adds part of the data. If somebody modifies part of the data. In addition to that, it also prevents complete false information. Right? So if we have a message to the other things before, they truly should not be able to modify the message in any way. But then at the same time, as a byproduct of that, if truly just sends a message involved, involved is not going to really believe that it came from palace either, right? So you can't even falsify the entire message. There are different levels of integrity, so different people also have slightly different views of it.

I guess the first basic view you can take of integrity is that you can detect accidental modification. In other words, you can detect error if maybe your background is in ee and that's a communication theory, some networking that people do error correction codes. The goal of those is, if I send data, it will figure out that there was an error in the communication. And also, then they will then try and fix some of this error. And that's very useful, but that's not really a security service. Because anyone that modifies the data on purpose, right? They will basically, also, they changed the error correction code to basically be correct. Right? So we cannot say the message has never been modified, because the error correction.

It's great because that's not a cryptic function. It's just an error detection function. Even until very recently, you would see things like this online. It used to be quite popular for like open ftp servers, and people have software online, sort of upload the software together with a hash function result. But as we'll see, like, their hash functions don't need a key to generate. So anyone can really generate a hash function. If I would go online, couple of dodgy file modified by me, I can generate a new hash to sit with it. I so that doesn't happen that often, but it was a habit that was quite popular a few years ago. The next step up is data origin of education. So we want to verify who sent us the message, and then it has not been modified. So if alice has sent me a message, I want to ensure that they say it was indeed sent by alice, then what that it was made by others, and then it does not be modified.

And then finally non repudiation, almost exactly the same thing. Alice sends me a message. I want to ensure that message is made by allison has not been modified, but more importantly, I want to be able to prove to somebody else that others made that missing. Right? Because if I cannot prove to somebody else and others made that message, then I cannot have that on reputation. If anything goes back and says, I have to send you this message. I have no defense against that. So we'll see a little bit later how that works. So essentially, we need a mechanism that only one person get it right. One thing that sometimes comes up around this time is that why do we need additional integrity service? If I just encrypt data properly? Would the receiver not see if somebody modified decipher takes? In other words, if the receiver decrypt the cipher takes, and then the attacker has missed the run with it, your plane takes is going to look weird. It's going to be funny. We're not gonna understand what it says.

A quick answer is, we did that tutorial where we did this streamside for encryption and the bank number, and somebody which is able to easily modify it. That's one example, encryption. And the way we use it might not provide integrity whatsoever, either what if we use, let's say, a nice box, I for the night mode of operation, and we encrypted box of message, and we sent it to him. Somebody along the line modify some of the cyber text that we know from error propagation that will change some of the main text.

And then bob can say the plain text looks funny. Ii don't believe that this is true. This is an interesting argument with the force flawed. In one way in that we are sending for data. Bob is not going to know what the data looks like. That's the entire point of sending it to bob. Right? So i'm sending bob a binary file with stuff. Bob receives it. He decrypts it, he looks at it. He's not going to be able to say this data looks fine, because he doesn't know what they've done sitting, or has no terms of reference for what it should look like. Only while you got. When we do with the education protocols in the next lesson, we will see that in very specific instances, encryption can provide data origin of education, and those are consensus with. Bob knows exactly what he should be getting in some protocols. That's the case because alice will above, for example, send alice a random number and ask her to encrypt it and save it back.

So in that case, bob knows what the answer should be. If the encrypted random number comes back and it's not the one that bob's in, he's gonna say something is funny. But generally, just for data transmission, it's good to have both confidentiality and its own integrity mechanism that can provide us the integrity service, because basic encryption things do not provide us with any integrity. Right?

The first thing we want to look at is electronic signatures. So electronic signatures provide as non recreation, because there's only one person in the system that we make the signature. So we could later say we've got a valid signature. This must have been made by the person who signed it. If alice signed it, I can also prove to other people that alice has signed it and that this message comes from us. Right? What are we thinking about? Is it is an electronic document that's going to be safe from alice involved, and we want a functional equivalence of a handwritten signature. Other words, it should be easy for others to sign. It should be hard for anyone else to forge alice's signature, but it should be easy for anyone else to verify that this is actually alice's signature.

We want the signature to establish origin of the message. In other words, we sent it as it be modified, and we also wanted to provide later dispute settlement. Otherwise, we want to prove to other people later that this message did come from us. And that is the non repudiation part. We want something that can be signed only by one person and something that could be modified. And i'm sorry, we verified by many people. What does that sound like to you? Does that sound like a symmetric system or asymmetric system? Prefer system? Because if I remind you, you remember that my public key encryption work, we had anyone can encrypt I one person can be cripple. Whereas for symmetric systems, it was sort of only one person can encrypt and one person can be correct. Would it sound more like a symmetric or symmetric? Right? The isometric seems to be the system where one person can do something, but another person could have lots of other people can do.

Another thing, right? And that would be correct. So digital signature is based on asymmetric crypto. The idea is that there's one person that could sign, so they're gonna use their private key. Remember, an asymmetric system is one person with a private key, but many people could have their public key. So alice could sign with her private key, which is called the signing key. Everybody else in the system would be able to verify alice's signature with her public key or the verification key that's happily distributed to everyone that needs it.

So in the signature, we're going to take the private key. We're going to take the message and we're going to sign it. Then we are gonna send the message with the signature. The recipient is going to verify using the public key, and it's going to decide if the signature on that data is valid or invalid. Right? So only the sign there can sign, but anyone can verify because everyone has a problem. One important thing to note here is it sometimes goes a little bit wrong. People forget when I can understand the recipient a signature, that makes no sense, right? The sender or the recipient also needs to know what was signed. Okay? Is that if I give somebody a contract and they sign it, and they just said, and then they cut the bottom part of the paper over their signature on, and they send it to me that doesn't help me anything.

Usually when we sign and when we do math, we send a message together with the signature. Otherwise, there's nothing to verify the signature on, right? Are there? Otherwise? It's pretty much useless. Right? Why did you do a digital signature? Is to use rsi the example that we get to look at is rsi at the end of the next year, we'll look at a couple of other examples, esa and one place on the street or the album also nature steam. Before we do that, we just remember again that rsa is a very popular algorithm. Then this so happens that rsa is this special algorithm that can both encrypt and deep and sign with exactly the same mathematics.

At the end of the lecture, when we look at other mall signature versus alcohol encryption, we'll see that it's very different. But for rsa encryption and encryption and signing and verification, the process in the entire of the system is actually exactly the same. Right? There are other purposes. There are other signature systems, like, as I said, is the album or signature is the digital signature algorithm. We just have to be a bit careful what we say. So in rsa signature, if we sign, we raise the message to the private key. Right? If we going to verify, we're raising the signature to the public exposure. Okay? So it's very easy to start saying we're interrupting with the private key, and we are decrypted with the public key, which doesn't make any sense. So we have to use the correct terminology, right? Because it might be true for rsa but it will not hold for any other signature scheme. It's much better to have terminology that fits everything.

So usually, we don't talk about encryption and encryption at all, and we just talked about signing and verification. Your assignment key is the private key, and your verification key would be your private key. What does rsa signature scheme look like? Exactly the same as rsa encryption? We're just using the exponents and the public private key in a slightly different way.

First, we have our two large primes, p and q and we multiply them together because n then we basically choose e and then calculate d that e is the modular inverse of e module of yn which is just p minus one times q minus one. The signing key would be d which is the private exponent. And the verification key would be e which is the public exponent. And in which is the public modulus.

If we do signature generation, the signature is message raised to be private exponent formula n and for signature verification is to the power e so our signature raised to the public exponent modulo in. We should give us our original sign message. If it's not the original sign message, the signature is invalid. Something funny happened to the message. Right? There is one problem with this. What is the largest message we can sign in this way? What limits the size of m if we sign? What is the largest value of a movement side? Any ideas? The length of y determines the length of n we can sign, I guess. It can't be larger than the modulus, right? We can't sign anything larger than the modulus. Like if we do the verification, we never gonna get our answer is bigger than eight. Right? It's how the math works. In the editorial, when I look at, we have one question that looks at how the encryption of rsa work works or why it works.

There's actually editorial questions is why can't you sign larger than in? Then i'll show a little bit of a deeper explanation of that. But if you think about it logically, if you did the signature verification, your recovered message will never be larger than m and this also implies to rsa encryption. I can't encrypt a message that's larger than in.

All right? Because if I decrypt, I will never get an answer that is larger than m i'll get the answer. Let's go grew into in, but it's not in. We are limited practically by the size of the mergers. So how do we solve this problem? Because we want to like basically sign very long messages. But now we are limited by a we could say we're just gonna break everything up and sign everything separately. But this does not work because this would be similarly useless like ecb where I could then take the message and preorder. Right? No one's gonna know if the original third block is not the first block, because it has a valid signature. We want a signature for the ims okay? The other solution we might use is we could make the modulus larger. Unfortunately, this is a bad idea, because the larger we make the modulus, the slower it gets at the same time, rsa as we increase the modulus.

And you need to increase the modulus actually quite a bit to make sure that it becomes more secure if we look at rsa and this is generally a problem with asymmetric crypto as a symmetric crypto. If I want equivalent, security is a matrix based on to forcing the key versus during the factorization. We basically see that rsi 1,024 would be equivalent to 80 bit symmetric key. Two thousand forty eight two hundred and twelve, 3,007 221, 28, seven thousand six hundred and eighty two hundred ninety two, and 15,380 to 256 pages. The performance of this is quite bad. Every single time you double the key, you basically go up q to the power of three in terms of how long it takes to calculate. If you add a system that you sign rsa one thousand 24 and 60, 30 seconds, which is quite realistic, actually, it was basically going to 620 seconds for rsa 2048. If you go to 15,000, 360, we basically take over 3 minutes to sign. Right? We can see because of the way the number theory works.

And the modular exponentiation we need to do is, however, Efficiently we can implement it, possibly the government, the numbers become very large. The calculation becomes very slow. Right? Realistically, we don't just want to make the modulus really large. We need something where we can so keep the modulus respectable, where we can choose the modulus basis guarantee, not on the length of the message. So in comes the hash function. So the hash function is sort of a helper function. So we'll see now how it helps digital signature. We'll see later how it helps message of information codes.

What we can do is that has function basically creates what is called a message, digest the hash function a it's not a unique representation of the message, but it's a unique enough representation of the message that it's unlikely that we can find another message that hash is to the same value. Right? And then what we will do is instead of signing the raw message, we would rather sign the hash of the message, send the message of the signature of the ball, and then we will take, bob will take the message. Bob will hash the message again.

All right. And then bob will verify the signature and see that the hash in its signature is received, matches the hash of the message that was saved. If these two things are fine, then false use that the message is correct. This heist functions made goal is basically to map a message or anything to a message representation or a message digest of a fixed thing. Right? Hash functions do not provide any security services by itself, because it has no key. It's an algorithm, but anyone could run it. Right? There's no special key or secret key. You calculate the hatch. It supports other mechanisms and by itself and did take accidental modification, but it cannot provide as data origin of indication, unknown repudiation by itself. It's like a helper function for our function. What are the requirements we have for hash functions? Hash functions has two functional properties, and three security properties.

The first function of property you should have is compression. So it should very effectively take a message of any length and method to a small fixed length. Output one is definitely going to be smaller than the modulus beginning of the signature. In addition to that, we wanted to run very fast, the entire reason for us could not use large modulus for rsa possibly was because of speed. If we want to compute the hash, computing, that hash should be incredibly quick. Basically, hash functions are as fast, if not a little bit faster than the block side. Right? Probably about the same speed, then hash function should have free security properties. The first security property is called one witness. If we are given the hash value y it should be impossible to find a message x that the hash of x is equal to y in other words, if we are given the hash output, it should not be able to be possible to find out what they include was right.

The next one is second pre image resistance. So if you are given a message why the hash value of that message y you should not be able to find a message x that the hash of x is equal to the hash of one. Then finally, this collision resistance. So it should be feasible to find any message x and y where x and y is not equal to each other, that the hash of x is equal to the hash of y at first glance, if we go through it, the first time, the last two seem very similar, but there's a big difference. From an attackers perspective, it's much easier to search for provision than to find the second free image.

The reason for this is, and the big difference between the two is it for second free image resistance? There's already a message and it has a hash output. I need to find a message that will give me exactly the same as result. Right? Whereas for collision resistance, from an attack perspective, I have free choice about all the messages. I have three choice to choose x and y and as long as I can find any x and any y where the hash of x and hash of y is the same, I would win.

So that is easier to do than trying to find a message that gives me a specific cache. Right? In terms of the collision resistance, and whether it is theoretically position resistant, or this computation is. So, it can never be theoretically collision resistant. Right? That basically means a hash function will never give a collision, but we know that cannot be true. If we're gonna have any function that maps, what should we call it an infinite set of values? Because it could be a message of any name. If we map it down to a finite set of cache values, there must be multiple messages that map to the site hash, right? Right? There's no other choice. If we say that we build a mapping function where we mapping 10,000 on the 1,000, then obviously more than one of the 10,000 shipped back to one of the 1,000 in cash. We have that text. If you're going from a any length value to a fixed length value.

In this, there's less possible digest, then they could be possible missing. It's just that it should be computational feasible for us to calculate what those two messages are very good, ok so it's basically something that's computation is secure at the end. Here. It is.

And then final, it's also easy, because there's no additional security dash function, anyone who wants to find a position and do so, because there's no heat, there's no secret thing that protects us from this randomly searching for english. It has functions in a way, is very similar to what side. First we think about them in similar ways. We write the messages up to the blocks. We then literally see these blocks into the hash function to get the final cache value. And it has similar properties to a dot function, right? Like if only 1 bit of the message should change, then the output of the hash value should significantly change.

So there's lots of now it has functions through the ages. You have basic charges four and before and five, the secure hash algorithm one. Now the secure hash algorithm two and three, 84 and 85 are not really used anymore. Show one is used in some specific cases that pretty much today when you show one. But then at the same time, they have already defined since 2012 and use your hash offer. But we're not quite at the stage with the hash on this. It's not efficient. So if you start, if we want to estimate the security of the hash and do something similar to when we have symmetric control, we can try to estimate what the efforts for the attacker would be. Do you try and find religion?

So remember, when we talked about symmetric encryption, we said the easiest way is brute force key. The algorithm is secure. Otherwise, the best way to attack it is by searching for the key. We looked at the program east fares, which is two to the nn with the length of bits. We said roughly an attacker always has to search half of that. Ideally, what we can do now is we can say, if an attacker wants to break the collision resistance of the hash, that's why we worry about the bugs, because that's the easiest thing for the attacker to do. Can we do something similar if the output is not? This time is not going to be the key size? This time it's going to be the output of the hash. If the output of the hash is 8 bits long, how much if it does the attacker need to basically find a collision? This brings us back again, then do some statistics and figuring out the probability and of success of the attack.

There is a related problem, which is called the free birthday problem. And then says that if there are, hey, people in a room, and how many people should we have in this room for there to be at least be a part of chance that somebody would have the same birthday as me, right? As we solve, it, basically find that we need, on average, 253 people. Right? There's 365 possible birthdays. Right? And what this problem says is I already have a chosen birthday. I can see that like the 1st of april, and if I don't want a classroom and I want to think about it differently if i'm standing in the corridor, and everybody that walks past, I asked if they did that, and I win when they have the same birthday as me. On average, I would ask 253 people before I find somebody that has the same birthday testing.

So this problem sort of sounds like it, but it's not the same. It probably sounds a little bit more like second three image resistance, right? Because the second three image resistance, we already have a message. We were trying to find a another valid message that matched that message, right? But we know that the attacker has free choice about birthday or message. So that brings us to the birthday problem. How many people must we have in the room before two of them have at least half a chance of having the same birthday. And as it turns out, that number is very small. On average, you only need about 23 people for one of them to have two of them who have the same birthday could be any birthday. There's no choosing chosen birthday this time. Unlike the previous one, we have a birth, then we try to find it, keep counting people. And we find, in this case, we say, I don't care what the birthday is. As long as two of the people have the same birthday, I would we only need 23, which is roughly the square root of 365.

The implication for hash function, essentially, that when we think about the security of a hash function and the effort to break provision of resistance, we basically say that the amount of work that an attacker needs to do is the square root of two today.

In other words, two to the n divided by two.

This is different to two to the n all divided by two, which is two to the n minus one. It's basically a square root of two to the n which is two to the n divided by two. If we have a hash output, that's 80 bits long, right? The amount of work we need to find the collision would be two to the 4th. Right?

So how is it different from the previous problem? Once again, if I stand in the corridor and people will ask me, I don't have a bit. They start with, I just ask this one, what their birthday is. I write it down and then I ask person two and I write it down, I write this in three, and I write it down, and I suppose it for, and I write it down. Right? As soon as I write down one, and I already have that birthday, I would quit in a way I have no frequency mentioned by what is going on. So ritual searching for hash is a lot of the same. So i'll randomly choose a number of messages. Every single time i'm gonna hash the message, i'm gonna write down the hash result. All right. And I will keep hashing messages, messages, keep passing messages. Basically, until I find two message values that are the same, because it's not a square root of two to the n instead of just two to the n minus one or two to the n all divided by two, it doesn't apply that hash functions, need a significantly larger output.

Then when we had what cipher heat, we must make sure that any hash function we design have a significant number output, right? Long enough in as.

Okay. So before we go to the break, I would like to do aa practical experiment. So I try this practical experiment every year. So this is the first year I could drive here, but i've always tried it in the other city members for the last 12 years. And that's to see if this actually works. All right. So actually, we had really good luck in this working. So for the first like 9 years, 8 to 9 years, I told the course it always work every year with the postgraduate student and undergraduate. And then for the last 4 years, it didn't work. And I don't know why, but I guess every single time you only have half a chance to get it right. So basically, but we also have more than 20 people. The fact, but then last friday, I tried it with the other campus and work very hard. What the experiment is gonna be is we'll start at the front, and the person will give me the first. And I will repeat the birthday, I will call you. And we'll keep worrying with a bit of that until we find a birthday that matches.

And then we can see how far we go before it matches. So we have 1/2 a chance of it matching early. Right? We can try, we can see if this is less than 365 people here. So it has to prove some of the maps. So we can stop ok so we can start here. What is your birthday? Ii a I need a month August. 1st. Anyone have an August 1st birthday? Excellent. The December 29. Anyone from December 29? Fine. December 29, then that's on the second one. Very quick, right? So we did the real 112345678, 67. We would have asked on the 17th, one.

Okay, so if we did, we like, I we do it slightly, because usually it takes very long. It's not very, very fast. That's very good. Right? So in reality, if I was doing a real high collision, I would write everybody down, and I would win when I get to a second person that has. This guy is who did you have one by less than 23? Right? In between they might have been another match. I don't know, but we would have definitely have one before we got to 23. Right? So you can see that it's quite easy to do, right? I think that's the process that they were matched, say again. So that's very good. Thank you. And I think that would take longer. Now you can have it. Right? So good. Okay. I will see it. It's about. Let's make it 15 minute break, so I will continue at 5 minutes. Possibly, thank you.