### Security\_week5\_Tutorial5\_2-20241008

说话人1 00:00  
That a opportunity to a distinct, this is digest, distinct, not unique, but distinct enough for it to be unlikely the two messages enough with the same value. Then we said we can estimate the amount of effort needed to find collision on a hash tradition being the easiest attack, finding any messages with the same hash value as being two to the n divided by two, where n was the length and bits of the output of the hash. Do we need spatial hash functions? Or can we just use other cryptographic functions we already have? So in theory, using a block site would have good hash property. Right? Because if you can make a way to sort of encrypt all the messages, all the message drops in your message effectively, then you should end up with something that is quite a nice representation of digest of the entire message.

Think almost like if I did cbc mode encryption in some way where we basically, there was something similar where we keep changing the changing the previous block to the next box right now without at because in theory, block cyphers give us quite good property.

Right? Small modification in the main text will be modification in the socrates. It is a very good see the random function. It's difficult to basically see the output, and they get back the input, all sorts of nice property. All right. So could we do that? And would it be a good idea? We could, for example, say we're gonna build a hash in the following way. The first intermediate hash value would be zero. Then we can compute hash one that is equal to the encryption using message zero as the key, basically has zero as the data that's being encrypted. And we just keep running it like this. Until we run out of message blocks, we use the final block h as the hash value. The length of the message is the multiple of the key sites.

Why should we not do this? What was the block size for aes anyone remember? It was 128 bits, right? Block size for anc was 128 for this. It was 64. The block size will determine the size of the hash value while a block cipher with the key of 128 bits, we wanted, the brute force that he was 2 to the hundred and twenty seven, which is quite a lot of effort.

If we're going to take a block cipher like as and try to build a hash function with it, and get a hash of length one, 28 bits, what is the effort needed to search for a collision? Two to the one? 2 to the 64 only, right? Because it's in divided by 2. And 3 to the 64 is not a lot of it. I said a lot of effort, but comparatively, it's not a lot of effort. We said people already searching for days in a couple of minutes, 20 years ago, the day's effort to find the key is 2 to the 55, right? Is that at 56, 50, 50 to the 64 is not very difficult. You do complication in one, and they prefer.

So the problem with this is that block cyphers, the block sizes are just too small. They were designed for something different. If we want to use it for a hash, it would be a poor candidate. Because the block size is too strong. You're digesting is going to be too short. So we're not going to go into a lot of detail when we talk about hash functions. But we can basically talk about some of the underlying architecture how it works. So in the past, and we take the first step we take is sort of in the past, the main architecture that people use was local doctor. So the first step is to take our message and split it up into plain text blocks. It is equal to the block size of our hash function. The block size could be markets.

Then we split up our message, and then our message is not an exact multiple of r we can have the end of it. So we have an exact molecule of then for merkel dot god, we said, we're gonna take an initialization, victor or initial value. We're gonna put it into the hash function together with p zero. It's gonna give us like hash one or hash zero. We're then going to take hash zero together with p one to get hash one. And we're going to keep doing this until we feed in all our plain texts, all previous scientific values, and the lost hash value that comes out of the system is then gonna be our message, digest our hash value, result in this design, the size that we feed in the size r always stays the same.

The amount of times that we do the hash function then depends purely on how long the message is not gonna message, the longer the hash function runs, right? And every single time we block stays exactly the same. When we finish everything is the hash result comes out. The hash result is the stated size of the hash bubble. For more modern hash functions, such as sharp read, they started using a slightly different design, so they call this a spanish design. The idea is just like a sponge. It can keep absorbing water or you can keep forcing it to absorb water.

And then once you are finished absorbing, you can choose how hard you squeeze it and how much water you squeeze out of it again. Right? These two different steps, the first one is absorbing, and the second part is the squeezing. So absorbing is basically taking the message into the hash algorithm. The internal state of the hash is size r flash size c your internal state that keeps updating is r plus c bits long. The size of r is variable one can choose how much of the message you will read it. Every round of the hash calculation. Right? You basically will keep doing this until you fit in all of your message.

Then once that is finished, you will start squeezing out that has result. Once again, you can choose how much of this data comes out. If you run contributes to your final hash function. This could also be size of r and you can keep squeezing out more and more. And the final hash function is basically the first output from calculating the second output. When calculated with not many other outputs you want to have. This design has various advantages. For the medical dunbar, we didn't really have any option to really customize it. You basically have to stick with a fixed block size of p zero being fit in, and the internal state, which contributed to the fixed link output, always have to stay the same. It was determined by the size of p zero and the size of the hash function. Or the sponge, it allows people to have a tradeoff. You can have one single algorithm, but you can have different tradeoffs between speed and security. The basic trade off is that the list of the message I read in every round and the more rounds I go through, the more secure the hash route would be going to be.

But the internet is slower, right? Because I have more rounds because I walk side is smaller. Alternatively, I can make my box size much bigger, which means I absorb the message much quicker, but then it might be slightly this secure. Similarly, with my output, I can make my size r quite small. That means I go through many runs to squeeze out a hash function that's long enough to be considered secure. Or I can make my hour larger, which means I squeeze out the required link of hesitate, digest faster. It's very easy to see this table between how fast I can process the data, versus I can go slower and make it more secure. It's actually quite interesting design concept. To summarize hash functions and signatures, a hash function is intended to take a message of anything and map it to a this is representative digest of a shorter fixed length that's gonna be definitely smaller than the modulus of the signature scheme that we are going to use.

So alice is going to take her message. He's gonna hash that message. This is going to give her the hash. They're gonna use their private key. She's gonna sign the hash. This is going to give her a signature. So it's just gonna send the signature together with the message to bob. Bob is gonna take the message. Bob is gonna hash the message. Bob, is it gonna take the hash of the message, the signature value and analysis privately and verify the signature. And then the result will either be true or false or valid or invalid. Involved and say the signature that I received matches the message that I received.

Therefore, I have quite certain that alice has made this message and that it does not be modified. Good. So it's not so popular. Crypto hash is. So as we said before, we have 85, and that's the 128 bit output that already something to be short. So people don't really use np five anymore. It is also have much more serious laws for parts of fun. We can have secure hash algorithm one. So it's similar to 85 and its internal designs that it has a hundred and sixty fifth output. It was.

Once again, I think the first government standard, just like sort of wait for encryption, from everybody using sort of that, I don't think there is as shower, like a sort of the first list driven secure hash algorithm. Subsequently, this also shot, too, so that it's rather, it also placed a short one, but the hash values are longer. Rather than having 160 bit output, you choose 224 to 56, 3090 4, or 500. In terms of the security of these hash functions. 55 had broken 20 years ago. All right? So some researchers showed that was very easy to find conditions for 85, right? But they could find many conditions in a very short amount of file. A year later, they also showed some attacks on shower one. They said that the effort to find a collision could be reduced to 2 to the 69, which is a nice academic result, but practically, maybe not that much difference in 2 to the eight. If you remember, shower ones, output is 16 dna the theoretical resistance we expect it is brute for searching for a collision would be due today.

The 2 to the 69 is less than 2 to the 80, but it hasn't completely broken it entirely. As a result.

Basically, 85 is not used anymore, and charles one is not really recommended for applications where provisions resistance is really important, but it's still likely fine in some other cases where we use it to build h max. We only care about second three and its resistance or one minus are just not for religions developing from 2000. And this recommends that shower line is not used. The position is this is an important aspect, the system for people to start shopping. Right? Just to get some practical context on that, introducing the effort from 2 to the 80 to 2 to the 69, what is the practical implementation of that? All right. It's so roughly 590 billion billion computation. Right? So even if you have some serious computing power, it will fail still take you a very long time.

For example, if you had a huge gpu server for, let's say you have some video of the hundreds. You had 1,000, 250 of those. They were all dedicated only to find a condition on shoreline using this specific technique that gives you two to the 69. If it would take you a year. Right? For practical purposes, this is not just gonna happen randomly, right? It would probably be an attacker that is quite seriously offer something.

If everybody last time we had an example of bit coin mining, we said that basically bit coin at its height at about the capacity worldwide have all the miners together. They could do due to the 65 patches per second. Right? If they are very costly finite position and show one, they will find one in a couple of statements. This is assignment to that. For good .. He's doing the group of work. You're not trying to find a condition. It's more relevant to to it's somewhere between one way news. And second three matrix is actually because what they will do is they will release a hash value. Then I will say, what is the value that make this hash? You have to keep searching for random numbers of a given link. Until you can find the hash of the message, it gives the given hash on. Once you find it, then you can have the big .. Right? So it's so slightly different collision problem. But if we were to do collisions, there would be one every 2 seconds. Right? So that is a lot of effort just for somebody to make a random attack. We also have to think about the practicality of this.

So one of the issues we can say is If we are using our hash function with a signature, and it's easy to find a collision, the drawback is that these two messages with the same signature, if i'm bob and I find x and y that has the same signature, sorry, it has the same hash function result, then I can sign x I can send it to others. And alice will say, yes, your signature matches for message ends. And when alice then comes back to me and says, I have to do what he said the message x and I felt her no, but I never send it to you. She says, no, but i've got a digital signature here that says, you send message x and then suddenly I send their message y and i'm like, no, I actually send message why? You'll find message y also has a valid signature outcome on that signature, right? Because the message x and message y gives us the same hash result. We are signing the hash, then the signature for these two messages always be the same, and that would be a problem.

So that sounds fine.

But now I actually also have to think practically, it is fine to go find random message x and random message y to give us the same hash value. But it's gonna be a little bit hard to find messages that have actual value. If I found others, I want to buy 100 computers. I want to make a second message that hash is to the same value. That is gonna be pretty hard. I would be have to be pretty small. Maybe somewhere in that email that I said, alice, I would have to slip in some random values, like a picture, or some random thing that I can change and try and find the high speed. This is not going to be just as simple as it would meet as I want to buy 100 computers. This is I don't want to buy any computer. Right? That's not necessarily going to be. You have the biggest problem. Actually, if I from, it may be being practical or not. And having an issue with the signature is more the reputational aspect.

All right. If I want somebody comes and says ive broken shore one, it's not easier to find provisions than it should be. Suddenly you have an issue in trust, right?

And then some funny things can start happening. Because remember, ideally, we want this to use this for non repudiation. Non repudiation means we have to prove that somebody was involved to somebody else, and that might involve taking them to court, pursuing them or making a case, right? And then if we get into some trust issues, the reputational issues, that can make it quite difficult.

So there's one story that illustrates this. So there was a gentleman in australia. He was driving along, he was driving too fast, and then he got caught by a speeding camera. So the camera took a photo in speeding. The way the internal of the camera work is that the camera would take the speed reading. It would take the photo of the car showing the license plate. And then we might do a digital signature of this data. It so happened that it used shuttle line as the hash would be the digital signature. This man then decided that he doesn't want to pay the ticket, which meant the police had to go to court to basically criminally prosecute him to try and get the ticket. And his lawyer came up and said, i've got this academic paper that says collisions on shoreline is not possible. Therefore, it could have been easy for somebody else to create this data and appear for it to appear to be a valid digital signature.

There's also the practical problems with this, but this was the legal argument. Right? And the judge said, given that this is a criminal case, and I need to convict this person without any doubt whatsoever with the burden of proof for criminal cases beyond any doubt, I cannot say that this person, for sure, is the person on the camera and he doesn't have to pay the speeding flight, right?

In a way that was quite smart.

To come up with that argument in a way the judge also doesn't really understand how a hash function works, or how digital signature works, because it's a nice job, but that is the top of thing that's going to happen. The moment somebody says it's not easy to do it, even the practice is not really that easy to do anything with it. Practically. The moment of trust in the mechanism erodes, and you needed to convince other people that it's trustworthy. You might have a ., right? Okay. One used to be a limited context and before, and if i'm not used anymore shock to being the one that most people use non practice, that said people sort of learned from as so for days, they waited too long, right? So they already knew they had some issues, but they sort of try to keep it alive. And then they came up with lots of variations before they finally came up. Yes. This time around, they basically said people are already starting to do things to show one and two md five. Let's just try and get a good thing for the future right now. We might not need it yet, but then it's ready to go.

In 2007, this launch a similar, aa stock competition to find the next secure hash algorithm or shot three. It was exactly the same as as they had a big, open competition to basically ask anyone to submit their design. They got 63 submissions. 51 was then selected for first round review. There were fourteen seventy five minutes, and finally, there was 55 minutes in 2010. Then similarly with as they then went to the finalists. And 2 years later they chose the winner, which was paycheck. Hijack is a sponge architecture, secure hash algorithm. It has variable link output, and it has variable throughput. Other words, you can have variable input and variable output, and you basically able to do this tradeoff between speed and security. Right? Currently, not a lot of people use it, but it's in pain, too. So at the moment, it sort of works with and not against sharks. So you can sort of use them together rather than replacing sharks.

Okay? And then the one interesting thing is that one of the people who designed it, so this was actually industry collaboration between ac market promise and speed. So one of the co designers with john norman that also happened to be 18 years. So we're going to shop three after the years years ago design. But right, so that's hash functions and digital signatures. It brings us the message of indication. This is just indication. This may be a little step down from digital signatures as it cannot provide us with another creation. But it is still important. Right? Is it just indication and data origin and indication wants to make sure that the message we receive is real and unmodified, but we might not have to prove it to somebody else. Right? If we think about some basic things, if we maybe think about some banking, so all the banks sit on a secure network, it's called the ship network. Right? There might be some interbank transfers or some payment safely read it. The confidentiality allergy is nice, but overall, the network is confidential and secure anyway, but your integrity is very important. You don't want errors, you don't want modifications coming.

And then data gets corrupted or modified, larger amounts get transferred. What do you want? Right? It's also probably unlikely that because all the entities trust each other already, that you're gonna really need things like modification. So in some cases, where alice involved is going to trust each other already, or maybe there's some other non abbreviation made measures like trusted third parties on the realization or transaction recording, you might have necessarily need particular signature. So you write like a slightly smaller, easier to use transaction that is faster to compute, to allow for more transactions. But you use the multiplication problem. So in this case, we can use message of indication. What is messages indication or the basis of indication code? I like the digital signature, the message of indication code is a symmetric graphic mechanism. The sender and the receiver both have the same secret key k the sender will take the message. They will calculate them back using this pk they will concatenate that to the message, and they'll send both to the recipient. The recipient will then take the received message, calculate the mac using its version of the secret key, and then compare its mac to the mac that was sent with the message and the two are the same.

Then we assume that the message has been modified, and the message indeed has come from source a right? But the message has been unmodified since source a has made.

If we compare it to a hash, is now already look at the hash of digital signature. If we compare it to a hash birth map, arbitrary long messages to a fixed length output, right? Are either a mac requires a key to calculate where is the ash? Did not. Anyone can calculate a hash. When we compare it to a digital signature, it is much faster to compute, because a it's a matrix encryption is faster than a symmetric encryption. Right? But the mac does not provide not repudiation.

Why does the mac not provide non reputation? Non reputation means somebody has done something. And I can prove that person has done it. So when they say they haven't done it, I can prove that indeed they have done it. The reason a digital signature provides non remediation is because there's only one person with the private key. Right? Is alice has private public key, gives everybody their public key. Their private key is always secure. She signs things with a private key. No one else should have a private key, which means that she signed something. It must be us. Or a message of indication code. Others has a symmetric secret key. Bob has a symmetric secret key. Right? Alice sends a message to bob. Bob can verify that anna sent him that message because she's the only other one with this key. It works from a data origin of indication perspective.

However, if alice now goes off and says, I never send you this message, what is in trouble? Because he can't go into anyone else and says message and send me this message with a map. That means alice must have sent me this message, because if bob goes to another person and tells him the story, they're gonna say, but don't you have the key as well? And then bob will have to admit that he does. So bob could, in theory, also have made up the message and calculated the macrophages, because it's in a symmetric system. We always have at least two people in the system that have this share key. Right? So we can no longer say only alice could possibly have done this. Alice and bob could not have done this, right? Saying there and recipient could have done this. So we lose the notification. So in general, the thing to remember is when we talk about unreviation, we cannot really provide it with any type of symmetric, because it's a matrix crypto. There's always multiple people with this, whereas an isolated crypto is only one person with a specific secret that provides the number of cash, right?

So how could we build mac algorithms? All right. There's actually two ways we can build cbc mac, or we can build hash macs. How does cbc mac work? Simply, we're gonna do cbc mode encryption with initial value of zero. We're gonna throw away all the intermediate blocks and only keep the last one. So we have a message, p zero, p one, p two, p three. And we have a secret share pk we can have c zero as the encryption of p zero. C one is the description of c zero and p one. C two is the encryption of cyx or p two.

And until we get to the final side textbook with the final plain textbook, then we're gonna throw ac zero, c one, c two, and everything that's on between. And we're gonna simply keep the loss of the fixed value as our mac value. Why does this mac implementation work? Because for cdc mode, the current ciphertext value depends on the current practice value, but also all previous practice values. If I have this hp zero, p one, p two, and p three, I calculated back across this by encrypting, everything with cpc mode. I only keep c three as my back value. I said that before and along the line treaty comes and changes, let's say, p one to x when bob and does the recalculation of the back, c zero is the encryption of p zero, c one is gonna be the encryption of c zero and x this is another one, which means c one is gonna be different. But c two is dependent on c one and c two. So c two is gonna be different, and c three is dependent on c two. C three is gonna be different. The mac tag that bob is going to calculate is going to be different.

Cpc mode is very good at detecting whether everything in the anything in the message to be encrypted has changed, and any change in the message will propagate. Right? Until the end. The one thing about this is the basic way that cbc mac works. Right? In reality, it's a little bit more complicated than that, because there's one attack where you can basically stitch different messages together. If you do it only in this way. Normally, if we just do it like this, and we can have mirror variable link messages, we are leaving ourselves a little bit vulnerable to combination effects of speech. In fact, so in real life, if they do cbc back, they would basically do cbc encryption for everything.

And then after they get the final block, they will do one additional permutation that additional permutation result. Will they need the back value? What would happen if we just use cbc mac? Professionally, then we can do the following. If the attacker has two messages, the attacker has two valid messages, and two valid back values for those two messages. The attacker can basically combine them together in such a way, the valid mac value for the new combined message is equal to the mac value for the second conventional basis, legitimate basis.

How does this work? If we have one message, which is p one and p two, and we have a second message, it's just p three, then anyone can basically make a new message, which is p one, p two, p three, x or to the macro value for p one and p two. And the mac value for this new message, would there be the mac value for the original second message?

How does this work? If we calculate the mac value for the first message, c zero is the encryption of p one. The mac value is equal to c one, which is the encryption of c zero, x or to p two, right? From the second message, the mac value is to c zero. It's the encryption of e three. Now the attack value, the tag message is the two sort of combine p one, p two, and p three xo to t dash. Right? If I send my modified message off to a valid recipient to validate the mac, the recipient is going to go. C zero is the encryption of p one. C one is the encryption of p two. And c two is the encryption of c one, x or to p three, x or to t dash, because this is what I said, my third law to be as an attacker.

But what do we know about c one? C one is also equal to t dash, right? The c one is the back value for the first message, which is d one and p two. We basically now end up with the encryption of t dash, xo to p three, xo to t dash, we x or something to itself. It becomes zero. If we export something to zero, it has no effect. So basically, the two key dashes disappear, and we are left with three. And the encryption of p three is basically equal to the second fact that e double dash, therefore, it is equal to t double dash. So the math tag for the third message is just equal to the backpack with the original second part on the basement.

That means without knowing the key or anything, i'm managing to provide two different messages together. By just saving the leaping block to the first block of the second message takes off to the fact that you of the business. That's why we normally do the final permutation here. So after we calculate this, there would be one more raw operation. You try for basis. Right? Why does this work? Because by choosing on leaping rock in this way, we basically make the system forget about everything they went before. We basically done cbc one with p one and p two. But by the time we get here to calculate c two, basically, we are only working with p three. In theory, for cpc mode, c two should be dependent on p three and p one and p two. We now essentially trick the system into forgetting about p one and p two, and also only working from p three on it as ap three was the third. This is wrong to be sick.

I would be like a more longer example of this in the tutorial for a dimensional file. Okay? The other way we can build a map function is using an h map or a hash function. What we will do here is we'll basically take the key value. We'll concatenate the message, and we'll calculate the hash of that. And we'll take that hash. Result will come calculate it with a second key, and we'll hash it again. Okay? The architecture shown here is sort of k times kx or to new pad and kx or to ipad. But this just means that the two keys are different in a system where we have one single key. It's basically doing a variation of the key to make ah mac key one and doing a variation of the key to make a ah mac key two.

So we can just as well say we have the concatenation of key one and m we take the result the president into t two, and then we asked everything trying to find ah map in h mac is essentially a hash.

Then we've introduced at value into for ah mac. We can also call them, some people can sometimes informally call them a key mac. Sometimes you will see a hash of the key with the message. H mac is two west. So it's feature that. So first we have the key within time message. Then we ask the hash output with abt right? We've talked about integrity and confidentiality. We've now talked about back functions. Hash is digital signatures to some final comments. We talked about integrity and confidentiality and doing both together. So what is the implications for this? How would we do it to encrypt the data first and then do the integrity over the cipher text. So we encrypt them back, or should we rather do the math? And then the equipment. So this is a little bit of a debate, and I guess it comes down to implementation issues.

So from a purest perspective, encrypted mac probably makes that where I work like, it makes more sense from a logical perspective. You say the mac should be on the plain text, and then we'll encrypt the bank takes together with encrypt. Right? I've heard of the reason other people say, okay, even though it makes sense to say the mac is really ensuring the integrity of the context, which you'd rather mac, the ciphertext.

So there's some benefits to this, such as if I do mac and encrypt, I first have to decrypt everything and then check the integrity. If the integrity that fails are basically wasted my time doing the description. Right? Whereas for encrypted mac, I can check the integrity. First, if the integrity files are not even bother, you think of. Right. We also have to think about how this practically happens. Let's say we use a mode of operation to encrypt, and then we also want to calculate the map. The implication is that we're gonna have to go through all the data twice. Right? It's actually using cbc mode to encrypt.

Now. We are gonna encrypt all our 100 lost. They'd say now we have outside the text. Now we want to calculate the map. Now we have to go through 100 blocks again. With a different key. The mac encryption key is not the same key. Right? And we only keep the last one. We have basically had to be forced to work through the data plus. So there's also a question about, is there any more efficient way to do that? So I think generally, and some people have showed that it's up to the base and how your application specifically work. But in the past, they had the some issues related to mac and encrypt. And some people think encrypted mac is a I don't think there's an exact science too hard. This work. There are better ways to do both at the same product. One interesting attack that comes from the implementation of the encryption. And the integrity on one single message is called the panic oracle attack.

The panic oracle attack assumes that we are using cbc mode to encrypt, and also cbc that implementation. What basically happens? Remember within that, if we decrypt in with cbc mode, what we are doing is we are decrypting, the lost sockets package. We have c zero, c one, and c two. If we decrypt c two, that's gonna give us a value, and we have to x or to c one to give us our plaintiffs. Right? From an attackers perspective, they already know c one. If an attacker can try to figure out the value here, this is gonna be, if I can figure out the last part of the value here, that is great, they can exhort it to the last bit of c one to actually find the plaintiff. Right? But how could they do this if they don't have the key? So when studying some of the common implementation of I think it was, this is our us they found out that in true computer science form, the people who coded it wanted to give people as much information as possible.

Actually, what the libraries did was it gave you a lot of technical information when it said it was an error. If I try to decrypt data, rather than to say your description failed, it tried to give me a reason why the encryption fail. One of the reasons that guy was actually that there was a back padding error. In other words, your padding box at the end for your back was incorrect. Right? And then somebody came along and said, but actually, this could help us find made it. Right? How does the padding work? So the standard padding standard that most people use is pkpkcss seven. It says if i'm adding 1 byte, it's 01. If i'm adding 2 bytes and 02, if i'm adding 3 bytes at 03, then I have 030303. That I think is that if you can basically pull the system, if you're checking whether the pattern is correct, it comes in. The padding is not correct that this last 5 year is not equal to 01.

So you're only looking for the first part. So the attacker when feeding it to the person who's being the description, it basically play around with this last bite of c one. The idea is if you play around with this last bike of c one and put it in different values, because this value here is always gonna be the same, the value that the crips here is always gonna be the same. We're trying to figure out what it is.

And now we have a choice to vary the last part of this e one until the value is 01 is that the value is 01. It's not going to give us an error. If the value is anything, but 01, then it is going to give us the error. We can keep this changing this. We can we can run through the 256 possible values of the bible. Not that we don't get a padding error, then we know the value that we have a sort of this value that we don't know is 0 points. 01. Once that happens, we know what this value is. We can exhort it to the real value of c one to find, basically the last plate explorer of c of b two. Then we can do exactly the same for the last two blocks, for the last three blocks, for the last four blocks. And eventually we can read up the entire plastics, basically.

So then I went up and fixed it. And then about 10 years after that, people realize that the problem hasn't really gone away. It's just that now you don't get a message anymore, but they found out if you fit things into somebody encrypted the data and waited for the response, some responses will take longer than other responses. There's something similar to happen. Remember that you would be able to see the back guidance fail, because the error response to 0 ~ five, 30/2 to come back. Whereas if it's correct, it takes ten, 30/2 to come back. They could use this timing analysis, find information about all this time, right? Now I don't think people have started doing is that we don't necessarily have to encrypt and then do integrity separately. There's an entire different family of mode of operation which allowed for something called indicated encryption. In other words, rather than do encryption, and then back, there's basically two encryption operations per block. We can use it as indicated, encryption mode of operation. It does both at the same time. The example here is gcfa broader counter mode.

It uses basically counter mode encryption. And at the same time, it calculates a MAC value. You only have to go through the data once. Right? You can see at the Top when we start with a counter and we increment the counter and we encrypt it, and we accelerate the plaintiffs to get cited text. We increment the counter, we encrypted, we extort the main things to get the side effects. That's basically the counter mode we looked at in lecture two. But at the same time, we encrypt counter zero. And then every single time we have cyber text, we start off with some authentication data, like an initial cash value or initial back value. We go through a limitation. We explore to see why we go through reputation. We explored to c two until we run out of cyber text. We do a final two permutations. We explore it to the first block, and this becomes our big patient path. Rather than I think the patient tag is like our backpack.

Later, when we talk about tls we'll see that in the later cyber suite, what the ls version three, all there are all crypto operations are basically not all invented encryption. Then you get encryption. Is this the mode of operation for a box cipher that allows for the integrity check and the encryption to be done at the same time?

So they finish off just the final note on signatures, because I realized 2 years ago that some people I want to ask about this, I just put it for interest.

I said that we would study rsarsa is quite unique, because the bats for the signature and the encryption is the same. In the last lecture, we looked at alcohol encryption. There is an alcohol signature scheme, but it is not the same. It also relies on dispute over the problem, but it is different. I just want to make clear what the difference is. Right? For rsa we said the signature was the hash rise to the private exponent. Modulus is and to do the signature verification, we did the signature to the power e which is the public exponent module a and this should give us our hash value. The actual hash value. This is another thing about rsa that's slightly different, and that's where they encrypt. And the signature is the same. Rsi is an example of signature with message recovery. In other words, if you verify the signature, your original thing that you sign will appear again. Right?

Let's look at some other examples. This is how the ball signature. There is a private key x there's a base Ga prime, PA random k and your hash value. You're gonna calculate the public key as y is equal to g to the x mod e r is equal to g to the k mod ps is equal to the hash minus x times r of the modular equations of k modular p one. Your signature has two parts, r and s and then to verify, the recipient will calculate the hash value, raise g to the power of the hash modulo e and then calculate the second value y to the r times r to the s modular p and if these two values match, the signature matches, right? But unlike rsa the hash that the sender has signed does not appear.

Again. The recipient only has his version of the hash. This is the same division of the hash. This is the receivers version of the hash to take outside. They use it in the calculation, but they simply get two values. And if the two values are the same, the signature is correct. For dsa similar y is equal to g to the x mod p r is g to the k mod p mod QS is equal to the one to the inverse of k one to the q times the hash of n times x times r plus x times r hash of in here is the hash message to the sender as calculated. And then your signature has two parts. And then the verification is actually quite tricky. So w is the modulator is of s modulo q u one is not a hash value that the receiver has calculated times w module q u two is r comes down to the modular q and then finally, we calculate b which is g to the uy times y to the uyypyq if this b is equal to the same r then the message matches the signature.

This one is where the signature doesn't actually give you the original sign value up, but rather have two values that you calculate. It's supposed to match is called signature with opinions, profit. So that is clearly for interest. Just because I think previously I used to ask a question, something like, could you use the album all significant encryption scheme? We let the next report to sign a message. A lot of people would say, yes, because they heard about alcohol signature. I basically added the slide and can just show them that it doesn't work that way for rsa is very special because the maps for the signature encryption is the same for other things, even though they call the same thing, alcohol signature versus alcohol encryption, the mathematics actually look very different. Right?

And also, the second thing that's interesting is that rsa when you verify, you get your original sign message up, but for other signature schemes, you actually done. All right? I think it's the only real mainstream one that does this. For the other ones, you have to calculate values. You never see what the original message was. You have to match the two values these years. That is the end of the lecture. If you are staying for the tutorial, I will start going through the solutions in about 30:00 an hour at 2:30, 4:30. Right? If you, in the meantime, have any questions to ask me about the lecture, you're Free. I'll be at the front. And all the notes and everything will go on to kind of. I understood that campus is gone for the last week. It wasn't very useful, but hopefully how it's back.

Actually, you managed to get the problem set before the camp was gone. I have, yes, managed to get it somewhere else. Thank you for the lecture. And then i'll go through the solutions for the tutorial enough to the 18. So it normally would be, okay, there have been switch ensure that it wasn't even more than 100 ~ 1 if we don't break dash, that is big, but not to get out. So just really, from the fact that box under that's more blocks out, they will make a good hash function. You read it is set for the religion of the.