Foundations of Cyber Physical Systems
Prof. Soumyajit Dey
Department of Computer Science and Engineering
Indian Institute of Technology – Kharagpur

Lecture – 54
Attack Detection and Mitigation in CPS (Continued)

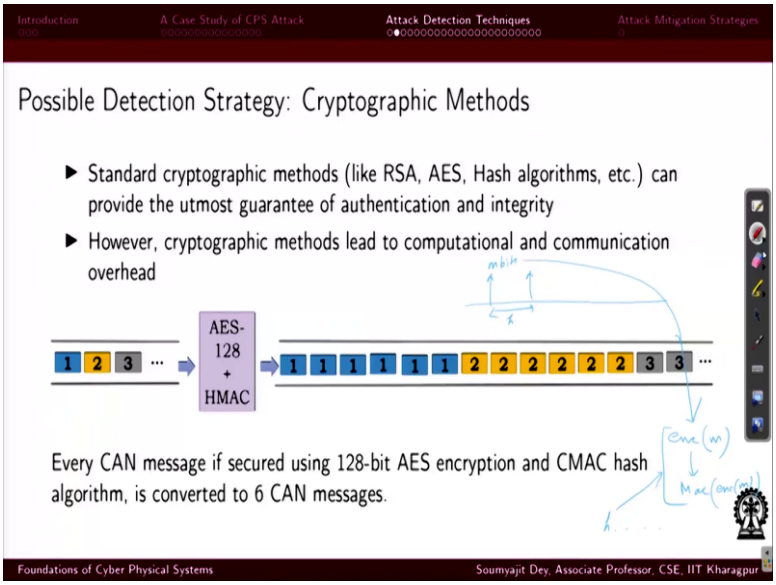
Welcome back to this lecture series on Cyber Physical Systems.

(Refer Slide Time: 00:33)

So, in the previous lecture we have been talking about ah some possible security issues in cyber physical systems. What can be possible attack models and how the attacks are modelled mathematically in terms of perturbations on variables y and u. And more importantly, we also showed the example of a practical attack that how the CAN bus is really vulnerable. And how attacks can be implemented ah to take offline, some some specific ECU.

So, what is important to understand is attacks will keep on happening on cyber physical systems because they are of real value, ah to the society, infrastructures and everything. ah So, ah such critical infrastructures need protection. right I mean not only infrastructure, so, real automotives and other systems, they all need protection. So, how can attacks be detected?

(Refer Slide Time: 01:23)



Now, of course, the strategy that will ah definitely come into everybody's mind is that well we have so much mathematically well known ah cryptographic methods for data protection. Right So, you have algorithms like AES and advanced encryption standard. You have RSA some hash algorithms, etcetera which give you Ah this guarantee of authentication. I mean for authenticating a data that is received from the sensor and also to check whether the the to take the data integrity, whether the where the data has been perturbed or not. right So, what can be done is you have a measurement, you first encrypt that measurement with let us say, ah 128 bit AES kind of crypto algorithm. And then ah this measure this data you also ah to protect the integrity of the data you add a message authentication code or CMAC using a CMAC hash algorithm. Right

So, ah you can add AES and MAC together. So, with AES you have data authenticity because ah on the receiver side, ah the data can be decrypted. right And the original data can be obtained. So, you know that well who is the sender of this data and also using the MAC ah you can actually check that well the data is ok or not. I mean ah because ah even if you encrypt and decrypt data, it may so, happen that the data is perturbed. Right

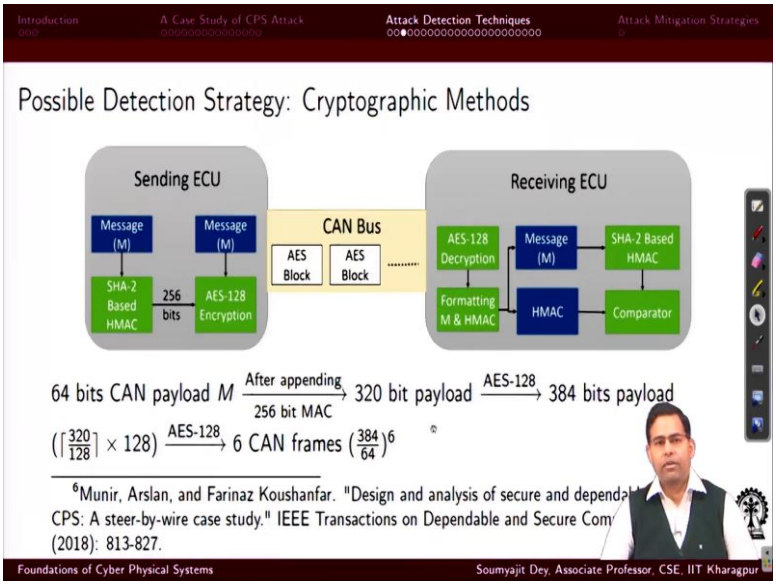
So that is why we will also need the integrity check and the integrity check can be done by an HMAC algorithm ah some some message authentication hash algorithm. right But what is the problem here? The problem is doing all these things take up resources. You need to run this AES algorithm and the hash algorithm at the same speed as the period of the data if you want to actually ah encrypt every packet. Right

So, if H is the period with every ah we inside every interval of size H. Let us say this is your data stream and you have a data point here, a data point here. These are all M bits and this is the interval H. That means you are adding an additional task of encrypting this M bits and then doing a MAC on that right this original encrypted data. And then this has to be done once every H period. right So that is a lot of compute that we will need to do.

So that will increase, that will increase your the you that your computational footprint on the low power, real time systems which may not allow that. And more importantly, ah your communication overhead will increase a lot. So that is the computation overhead we talked about and now what about the communication over it. So, it can be shown that if you have a CAN message and you do and a 128 AES encryption.

And this HMAC comparison HMAC base algorithm for a standard payload size of a CAN message. Then the amount of bits that will be in effect you need to ah transmit. Ah I mean the I mean, since your number of bits will now increase due to this AES encryption. right So, effectively now, for every original CAN packet, you will need to convert them to 6 CAN packets. So, it is not only that you have to invest resource for doing the computation, your message bandwidth needs to be significantly more. Right now that may not be an acceptable solution for a low resource, real time communication and processing. Ok

(Refer Slide Time: 05:19)



So, here we have an example where we are trying to show that it at what conditions this will happen. So, let us say you have a 64 bit CAN payload message M. So, these are sending ECU. So, it will ah implement this ah SHA-2 based algorithm ah for a for doing this message

authentication. OK And then this will generate 256 bits ok because ah your original 64 bit, CAN you have to append this 256 bit MAC Right and that would create these 328 bit payload. OK um

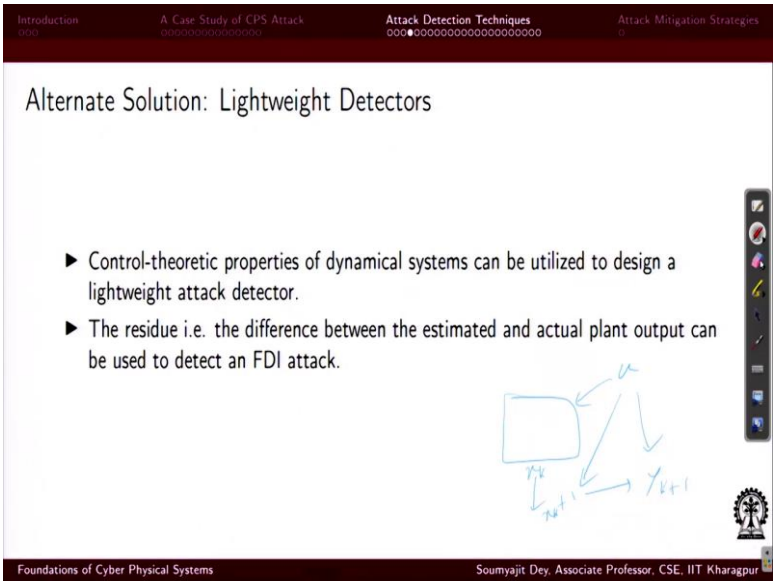
And on this you will, if you will run this 12 and AES 128 algorithm, you will effectively have ah this 384 bits payload. Ok Now, ah if you consider, ah this CAN packet size to be 64 here. right Ah Then effectively, what you have is something like this. So, you are running AES 128 so, effectively for let us say you have you have this 384 by 64. It is effectively Ah 6 CAN frames that you are kind of generating. So, ah you have on the CAN bus you have to send this AES blocks.

And not only that on the reception side, you will have first a decryption task. right And then after doing this decryption, you will have the message and the message authentication code together. You will you will separate them out and then you will do this Ah hash algorithm, ah copy of the of the hash computation. And then we will check whether the hash matches with the original HMAC. Ok

So, all these new tasks have to executed and not only that the original CAN payload which was 64 bits Ah 64 bit CAN payload. Now, this ah would result to this the I mean you you will essentially have ah this after you after you append this 256 bit MAC. You will have 320 bits payload here and once you do this encryption, you will have a 384 bit payload here. right So, ah with this 384 bit payload now, when you are transmitting this over the CAN bus, you will have to divide it into multiple CAN packets. right Ah

So, essentially, you will have this 384 divided by 64 right because that is the size of your CAN packet. So, effectively you will have this 6 CAN frames here.

(Refer Slide Time: 07:56)



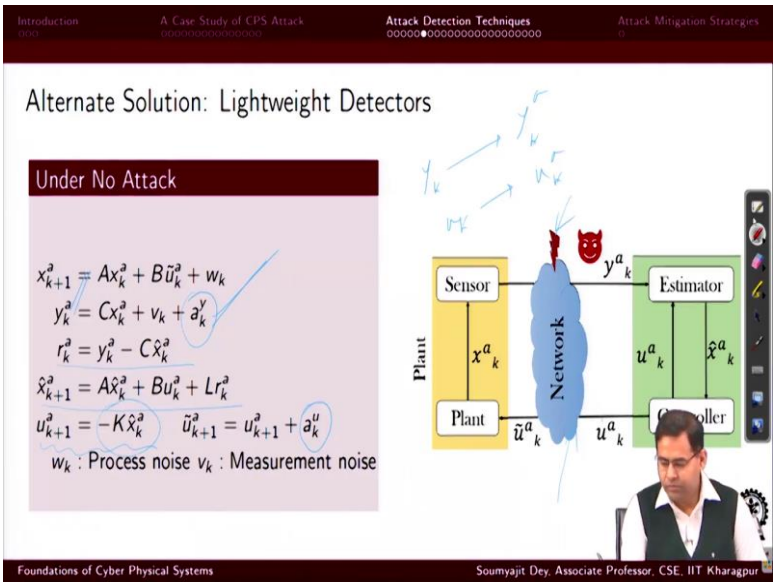
So, ah we understand this cannot really be a solution. ok So, since this cannot be a solution, ah what can really be done is you can actually use light weight statistical detectors which will try to correlate the value of different messages and see that well there is a strong, there is a strong correlation. There is a strong correlation or not. ok Now, for building such light wave detectors, what we do is, we will exploit the control theoretic properties of dynamical systems because we have one weapon here which is that we know the model of the dynamical system. That means we know that well if we give this input ah let us let us let us talk in a very simple manner. Let us say this is your plant and let us say this is now your control input. And right now your plant state is x we know that well x could go to x plus 1 Ah or some some operated state.

And from that I will get a y_{k+1} and we we can actually estimate that well what I am expecting x to be and what I am expecting y to be. ok So, this is x_k, x_{k+1} and y_{k+1}. I know that well for this x if I mean based on the current measurement that I observe, if I give this much value, what do I expect other output. I have a idea about that based on my estimation, based on my knowledge of the physics of the system.

For example, if I have a if I if I have a vehicle driving at a speed and I give this much of a break, I know that in the next cycle, ah the vehicle speed must be lesser than what it is now and it should be lesser by maybe maximum of this amount. I can estimate that. Right So, I can use this kind of mathematical estimators to figure out well what should be an expected value and then I can correlate it with whatever value I am observing.

And I can see this difference between the estimator and the actual planned output. And that can be used to detect whether there is a false data injection type of attack or not.

(Refer Slide Time: 10:01)



So, if we just put all these things in mathematics. So, under no attack scenario, this is your system right x_k, y_k is the estimator output x_k hat etcetera. So, this is your equation of the system update with process noise. These are equation of measurement with measurement noise and you know that well there may be a difference between this estimation and whatever is my measurement so, I will have this residue. Right

And based on ah the Kalman filter gain L. I will have this equation for the observer system. And I will have this equation for controlling for the control input computation. ok Now, when there is an attack, what is happening is, ah we will we just replace these original variables with their attack variable variant. So, what we have is let us say this ah an amount of attack that is happening on the system. Let us lets start to characterize them.

So, we will say that let us say on y the attack is modelled in this way mathematically that ah original unperturbed value of y is y_k an under attack we call it y_k^a. So, we are putting in a superscript on every variable that is either directly attacked or that is getting modified due to attack on other variables. So, let us say the direct attack is happening here. And similarly, on u_k we have u_k tilde right under attack, u_k^a, that is how we will write it.

So, so these are the under attack variants of the variable names. And now, we also need variables to model the amount of attack. So, we model the amount of attack that is really taking

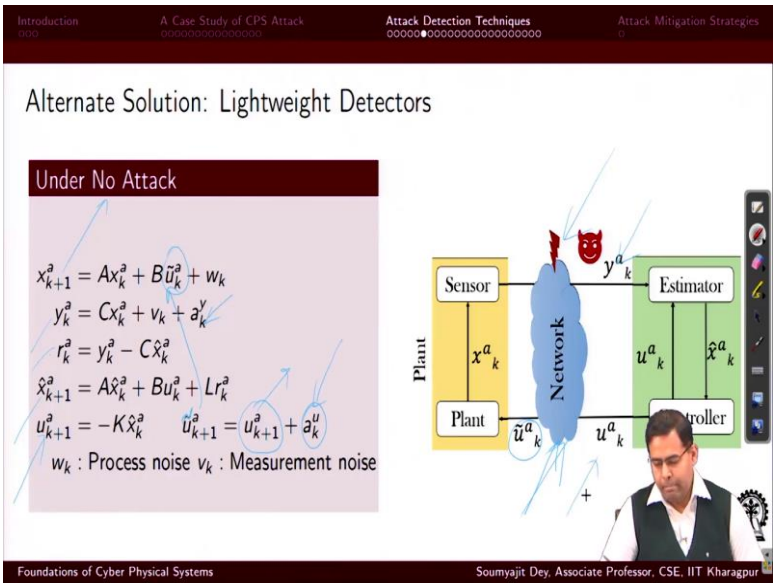
place on the y and the u using these two quantities. So, let us understand there is some attack happening here. There is a matter happening here due to which these values y's and u's are changing.

And that is why we we add this attack subscript but we also need to tell well how do I model the amount of attack. So, fine that can be a total replacement of the whatever is the current value but the way we model it, as is an increment of the current value. I hope that is clear that what I am saying is well the attack need not be a perturbation over the current value. right If that is the case, we will have the current value plus some unknown value added.

It can that can even be that I totally disregard the current value and add a new value. right So then what the mathematical way to model it would be that you set y_k^a to be 0 and then you just have that is y a this a_k^y which is the new value and that itself is the amount that the attacker is injecting. right So, when I do it mathematically, we are saying that the under attack, y is nothing but the original plus this increment that is being added as a perturbation.

And under attack all my equations are changing with this a superscript. This equation is also changing because they are all working with attract values. So, we are introducing that a superscript. ah The control input is also working with an attack value. right And then whatever is computed here ah it is superscripted because well ah let us see the sequence here. So, if we see the sequence here, ah let us go let us go through this once again.

(Refer Slide Time: 13:40)



So, we are saying that well there was some value that was injected here due to this y changed. Now, due to this change of y what will happen is ah my ah my residue will change due to this change in residue what will happen is my x hat will change. So that is why the a superscripts are coming in an interdependent manner. Right So, this is attacked, it is effect is felt here, that effect is felt here, now that effect will be felt here.

And so that is why I am superscripting this one here. And then ah since x has get got modified and u has got modified there is also an additional thing that can happen is the attacker can keep at perturbation here. So, this perturbation is modelled using this variable, a_k^u. right So, what we do is on the, on this attacked version of u you also add the a_k^u and then you represent this using this tilde. ok So, let us understand this mathematical symbolism.

We are introducing attack here in the y and the effect of that percolates through r x hat and up to u that is why we are superscripting r x hat and u with this variable value, variable a. right And then we are also saying that well not only u is superscripted with a. But only due to so this superscript is coming due to the indirect effect that y brings in, the a_k^y brings in. ok But apart from that you can also add a real attack value to the control signal itself and that is this a u_k. Right

So, this one has a superscripted to model the effect of the measurement perturbation, a_k^y. And due to that ah by this chain of events ah the the control input has got modified. And not only that on this modified control input, you can again perturb it further more because that is what the attack model here allows you to do because you can perturb here as well as here. right. So, you can add something additionally.

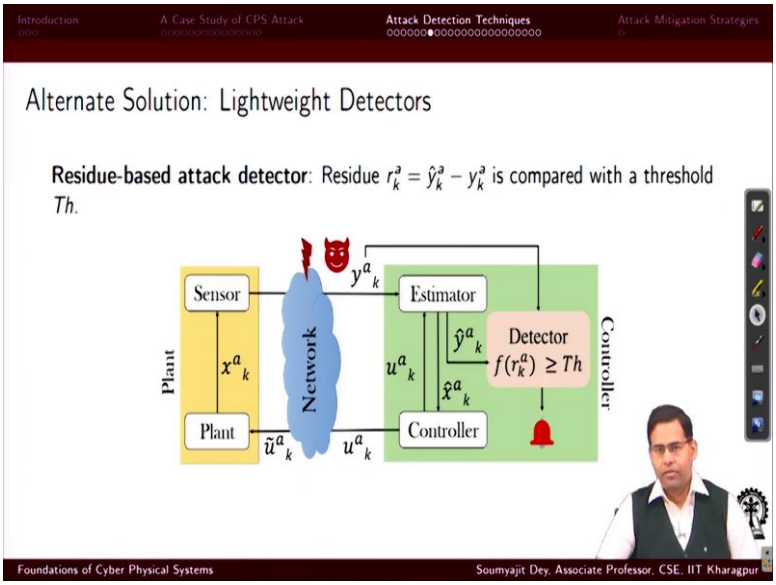
Now, if you add this then the output cannot have the same symbol. So, you add this tilde here right. So that is it. Now, you have this u tilde a k as the perturbed control input which is really flowing into the plant. I am repeating this as two perturbation components one is the disturbance that percolated, due to this a_k^y attack. Due to that y changed, r changed, x hat changed, u changed.

On this change due you further you can add this amount of attack here and then you get u tilde a y_{k+1}^a. And this effect will now go into the plants dynamics and that is why the plants overall

dynamics under attack will be affected by y in an indirect way and this in using this a_k^y and also with this a_k^u. ok So, I am just trying to tell why, on u I both have a tilde transformation and a transformation.

One is due to the indirect effect of the attack, a_k^y and also due to the direct effect of this attack u k. So, this is how I am mathematically capturing the effect that can come to the dynamical system ah due to this attack perturbations that can be injected both here and here.

(Refer Slide Time: 17:19)



So, once this happens ah we we, we have the following situation that you can add a residue based threshold residue threshold based attack detector. So, what it does is well you see we are computing, this right that is the residue under attack. So, you actually you you have an estimate. right you can based on your current observations you see your current measurement. Based on your current measurement you can you have u r this. Right

Based on your previous measurement of y ah you actually computed an estimated state based on. Let us say you were you you had an observation of earlier your observation of y_{k-1}^a. right So, based on that ah y_{k-1}^a u computed residue r_{k-1}^a. So, based on r a k minus 1 you have computed x a k hat. Right So, once you receive y_k^a you can use y_k^a here and you can use that previous computations x_k^a hat here and you can compute the residue.

So, this C x hat a k is nothing but y_k^a's estimate. right So, just like you have the original measurement you can have this estimate here. So, this C x_k^a hat is written here as y_k^a hat right

and you are comparing y_k^a hat with y_k, y_k^a. And you are seeing that well ah whether this difference is too much beyond some preset threshold Th. So, I hope this is clear. What we are really doing is we are having a detector circuit here which is which is looking at the current measurement. right and based on the previous measurement it has estimated Ah it is previous, ah it based on the previous measurement it has an estimate of the current state. And based on the current state it also has an estimate of the current measurement. right And it is comparing this difference between what it is estimates to get under attack and what it is actually observing. Right Right

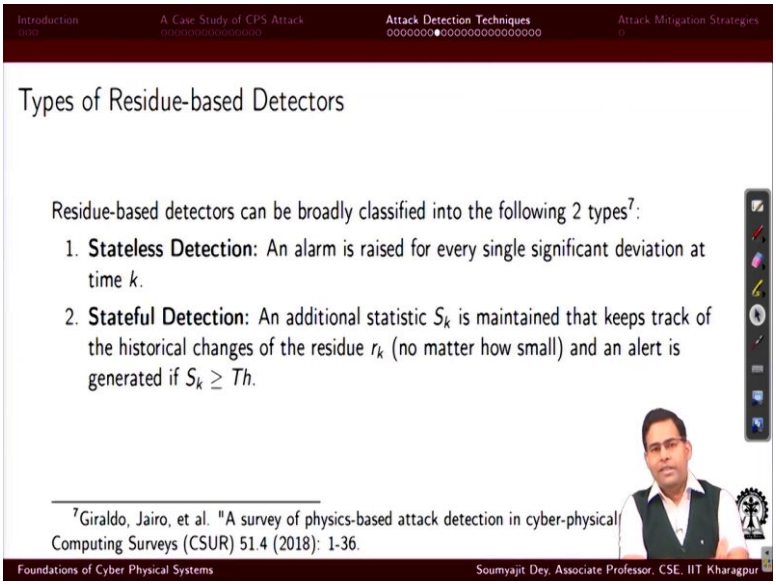
Now, say that earlier also there was this attack but had it been the case that in this cycle there was no attack. Then whatever I am measuring here right now and whatever I am estimating based on my previous attract estimate would be very nearby. Let us understand. This suppose there was this attack happening in the previous cycle and due to that I computed the residue but now in this cycle there is no attack.

So, ah based on the previous cycle's attack, I got some previous error erroneous measurement due to that measurement there was a residue. But due to that residue ah I also estimated what would be the state in the current cycle and this state ah will not be perturbed let us say in this in this cycle then Ah in that case, whatever I will get as an estimate of measurement for this cycle and whatever I capture as an actual measurement that difference will not be much.

Because I am considering that there is no attack. But if there is a significant attack here that difference will be beyond some threshold and this detector will raise an alarm and it will sound that well there is really an attacker present in the system. Now, you see that with respect to doing all these crypto computation that we did here ah doing this check is simpler. It is just a set of equations which are anyway getting computed here because of the estimator controller, etcetera.

All you are doing is you are computing the residue which is also a part of the Kalman computation here, the estimation here. But now, you just you use this residue in a comparator and you check whether this residue value is too much or not. So that is your detector right.

(Refer Slide Time: 21:11)



So, such detectors have actually got wide application in several kinds of CPS systems because they are very easy to implement and they may be effective also. So, it can I mean the detection can be of two types they can be also more involved. The simplest one is the stateless detection, where we will raise this alarm for every single significant deviation at the time k.

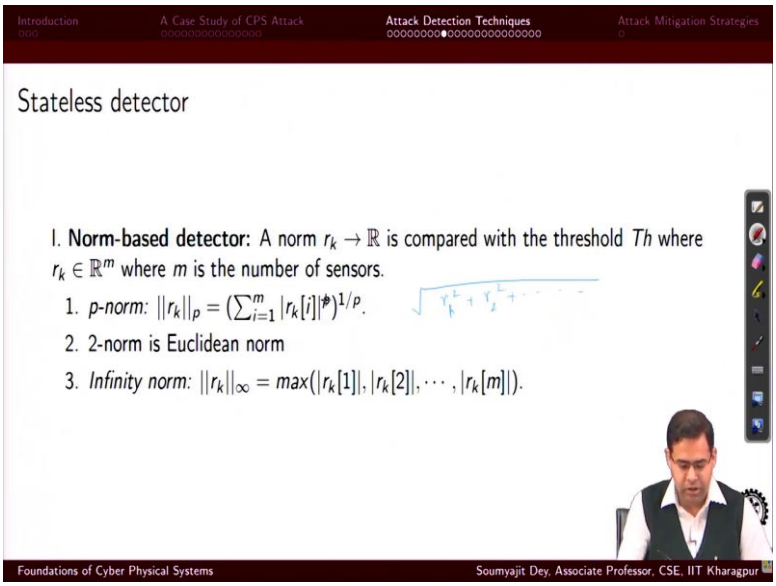
At every time, step k whenever there is a significant deviation beyond the threshold ah we will be raising an alarm. So, it does not remember the previous state or whatever. right So, it is, I mean so, in every cycle I do this comparison and if I see that there is an issue, I will raise the alarm so that stateless deduction. Stateful reduction means I will compute an additional statistic ah which gives track of what are the historical changes of residue.

Because maybe in previous cycles, the residue changed by a small value. ah but But maybe that was the attacker's intention. It is changing the it is changing the perturbation with very small values so that in every cycle the residue is not too much. So, ah the attack is actually escaping the monitor ah but but slowly, the dynamics of the system is shifting from some from some threshold from some set point. So that is more like a 0 dynamic attack with with continuous perturbations of very small values.

I am slowly diverting the control system from a state point to something somewhere else but nobody gets to know that. So, to thwart such an attack, what I will do is well whatever has been the earlier changes I am summing them up and maybe so some doing some is one possible way to keep track that there can be other ah I mean non memory less stateless detections, also. OK ah

So, ah let us say I am just using a simple technique. I am just summing the residues and so that that is one kind of statistics I can generate. And I will eventually compare well what is the accumulated residue if that is greater than a threshold or not. So that is like a cumulative sum or CUSUM detector. We will talk about that.

(Refer Slide Time: 23:23)

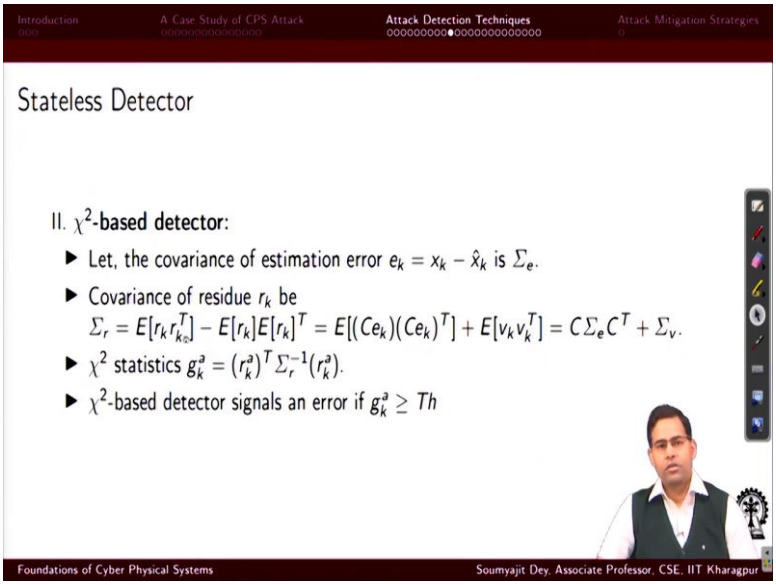


So now, ah for stateless detector, ah what we do is we compute a k norm right. ah let us say there are m number of sensors. So, for all those sensed values we compute the residues, OK and using using these residue values we can compute a norm. So, let us say it can be any p-norm which is computed like this or let us say it can be the standard for p equal to 2. What we have is the standard Euclidean norm.

So, I hope everybody can identify this equation so, just set p equal to 2 ah it is your standard Euclidean norm. right And for any other p for any other your choice of norm it would be the some some stand Some some ideal p-norm here. And you can also use any other matrix norm, for example the infinity norm the point is, you use some matrix norm to figure out, ah how we will compute the residue Ah as your base method of computation?

And then you will compare this ah residue with respect to the threshold that you may have chosen for your system.

(Refer Slide Time: 24:32)



So, for example, you can have this kind of a chi square base detector. So, what you do is you see the covariance of the estimation error. So, you you are computing the error right and you are let us say this error variable is e_k and so, essentially you let us say you are computing, the covariance of the residue. And so, earlier you just computed the residues norm and compared. But now you you see what is the residue sequence.

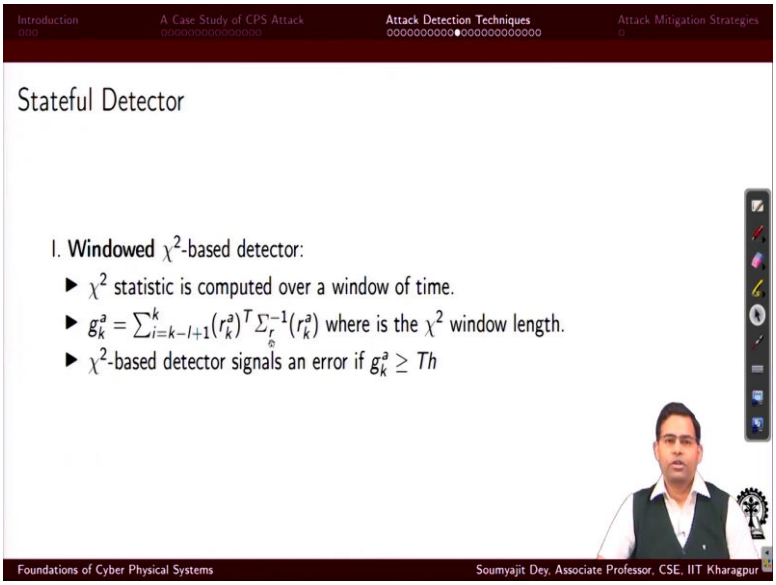
And for the residue sequence, you can check, what is the covariance through an equation like this. right So, we are assuming that for this error Ah you know it is ah I mean a mean and variance of that. right So, using I mean I mean in that way you can compute the variance using this kind of a formula here. right So, the estimation error is sigma e and you are just using this weight matrices here and you are computing ah the covariance of the residue like this.

And in that way what you are doing is you are computing, this chi square statistics ah of this covariance Ah r_k and let us call it as g_k. So, all you are doing is you are computing the residue and you are just using the covariance of earlier observations of the residue as a weight matrix here. ok So, this is your sigma r ah which is the covariance ah of this residue ok ah which you have computed from earlier observations.

And then when the system is deployed online, you are using this covariance as a weight matrix here and you are generating this chi square statistics g_k. As you can see that this formula is nothing but you are again computing a quadratic term here, ah with r_k^a transpose. Then the weight matrix which is the covariance matrix covariance of the residue, followed by r_k^a again.

And chi square will detect the signal ah to be erroneous only if this is greater than a threshold. So, this is one way one kind of chi statistics that you are using which is the chi square statistics. And you are using this for designing a stateless detector because again ah you do not have this idea of what what is the previous state of the residue. We are working with the current value only. But what you are doing is you are waiting this current value with this covariance ah that you have already kept derived in your system.

(Refer Slide Time: 26:58)



Now, for stateful detectors, we already talked about CUSUM right that is the cumulative sum. So, this is a way to compute the cumulative sum over a window. Let us say you are using an window of length l. And over this window of length l you are just summing up the residues like this. And you are going to sound that well there is an error if this sum goes beyond the threshold. Ok so ah

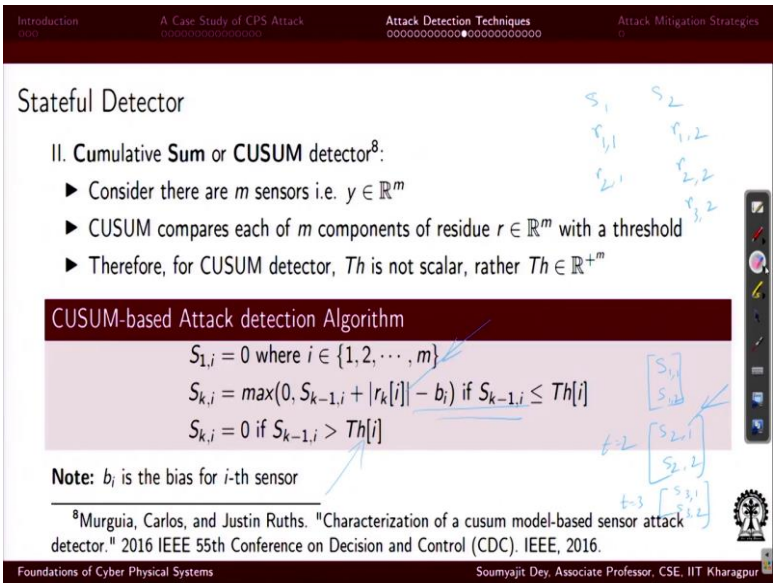
So, here you are actually using a windowed lambda square detector. ah So, this is not a standard sum we are computing. What you are doing is you have computed the chi square statistics here and you are doing a cumulative sum over the chi square statistics. But you are not doing it infinitely you are doing it for a window of size l. So, for a sequence of ah this chi square statistics. So, let me just do a small recap here.

You have got the estimation error. right And so, you have you know you have through this expression you have computed the covariance of the residue. And then you use this covariance as a weight matrix to compute the chi square statistics here. And then you are comparing the

chi square statistics with the threshold in a stateless manner. If you are doing a stateful manner then you do a cumulative sum computation but over a window of length l.

So, you are adding up this chi square statistics over this window. So, as you can see, this array is being multiplied with the weight matrix here and for each, for each computation of r_k. And then you are adding them up over this length window and you are just checking with respect to the threshold. So, this is windowed chi square base detector.

(Refer Slide Time: 28:45)



You can also have a simple summation computed ah over a window. So, let there be m sensors and ah what CUSUM will do is it will compare the m components of the residue with the threshold ah so, for for this detector. So, this is for a ah for a vector system so, this this the threshold is not a scalar but rather you have threshold over ah over over an m sized vector. And what you do is for in each ah for each of these vectors, you are computing, this ah state this state values.

Initially there was 0 and then what you do is ah in in inside this window you are actually summing them up. So, as long as this summation is below a threshold value for the ith component or the ith sensor, you are summing the previous, the partial sum that you have with the current residue minus of bias term. Ok Now, this bias term would be there so that these values ah are not too much I mean so, too much skewed because the sensor can also have a bias. So, to remove that bias for specifically for the sensor, you can have a bias term subtracted. And then this residue value you can keep on accumulating by doing this summation. But at the same time you do not want this summation to go on infinitely. Because then it is a

unmanageable system. So, whenever you are crossing the threshold that is one detection that is anyway happening.

So, after each detection window, what you are doing is ah you are You are resetting this thresh this Ah this cumulative sum for that ith component to 0. ok So, as you see that for k minus 1 in the previous window, ah it crossed the threshold limit for the ith sensor. So, in the kth window for the ith sensor, you set it to 0. And otherwise you just sum up the current residue in the kth window, with the previous sum removing the bias, of course.

And you see that will ah in case right now, my cumulative sum is less than the threshold then this is my then I am just incrementing the cumulative sum like this. So that is how you compute cumulative sum for each of these ith sensors in the sensor vector. And technically what you are doing is you are comparing each of them in a ah in a component wise manner. So, ah as an example suppose you have two sensors.

So, for each sensor you you have this thing that you are computing. right You have, ah in the first ah first instance $r_{1,1}$ in the second instance $r_{2,1}$ like this for the second sensor $r_{1,2}$ in the second window, $r_{2,2}$ like this, you are going. right And you are generating the statistics for each of these sensors using this formula here. And you are using, you are applying this formula as long as you are not crossing the threshold for that component.

And whenever you are crossing the threshold for that component, you are resetting this This statistics to 0. So, overall using the residue, you are building a window statistic where you have this $S_{1,1}$ and $S_{1,2}$ in the next cycle Ah t equal to 2 you are going to have $S_{2,1}$, $S_{2,2}$ ah like that. So, this way this thing is continuing. And what is happening is whenever one of them is going beyond the threshold then you keep on just resetting them. Right

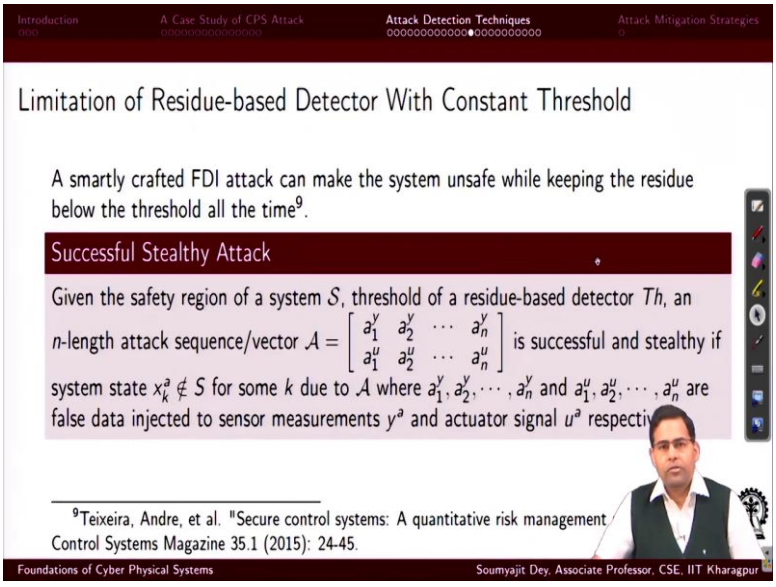
So, it may it is perfectly fine that in one cycle of computation one of the components get reset but the other does not because for the other sensor you have not compute you have not ah crossed the threshold, so that can always happen. So that is our cumulative some base detector works and it handles multiple sensors. So, for each sensor in that component, it is it is adding up with the previous threshold as long as you do not, do not cross over the threshold in that iteration.

So, this is how ah stateful detector works. And we also had this previous example of stateful detection here where it was windowed chi square. And earlier we introduced what is chi square at each case k step. So, all that meant it meant was that well you you have, you are ready with the variance of the residue. And you are generating the chi square statistics by observing the current residue.

And computing this term using the variance and then you are just comparing. And here you are, you are summing it up over a window of length l. And here you are just simply there is no there is no variance-based computation you are simply summing up the residues. But you are doing it for each component sensor you are tackling multiple sensors together in a vector. And for each sensor you are computing these statistics which is nothing but the cumulative sum.

And whenever any of those computations inside the vector crosses threshold, you are just resetting it. And in that way you compute keep on computing, the entire vector for all the sensors.

(Refer Slide Time: 33:58)



So, with this, maybe we will be concluding this lecture here. Thank you for your attention.