Ken Leonard: ITS Standards can make your life easier. Your procurements will go more smoothly and you'll encourage competition, but only if you know how to write them into your specifications and test them. This module is one in a series that covers practical applications for acquiring and testing standards-based ITS systems.

I am Ken Leonard, director of the ITS Joint Program Office for USDOT and I want to welcome you to our newly redesigned ITS standards training program of which this module is a part. We are pleased to be working with our partner, the Institute of Transportation Engineers, to deliver this new approach to training that combines web-based modules with instructor interaction to bring the latest in ITS learning to busy professionals like yourself.

This combined approach allows interested professionals to schedule training at your convenience, without the need to travel. After you complete this training, we hope that you will tell colleagues and customers about the latest ITS standards and encourage them to take advantage of the archived version of the webinars.

ITS Standards training is one of the first offerings of our updated Professional Capacity Training Program. Through the PCB program we prepare professionals to adopt proven and emerging ITS technologies that will make surface transportation safer, smarter and greener which improves livability for us all. You can find information on additional modules and training programs on our website www.pcb.its.dot.gov.

Please help us make even more improvements to our training modules through the evaluation process. We look forward to hearing your comments. Thank you again for participating and we hope you find this module helpful.

Nicola Tavares: Welcome to Module CSE 201 Introduction to Security Credential Management System, also called SCMS Part 1 of 2. Your instructors for this course are Dr. William Whyte is Senior Director of the Technical Standards at Qualcomm Technology Inc. Following the acquisition of Qualcomm onboard security where he was CTO, William is one of the world's leading experts in the design and deployment of security for connected vehicles and general mobile ad hoc networking systems. He is the editor of IEEE 1609.2 the baseline standard used worldwide for connected vehicle communication security, and of its related and successor standards. He was key contributor to the design of the Security Credential Management Systems for connected vehicles in the United States, and lead security consultant on the New York City connected vehicle pilot deployment.

We also have Dr. Virendra Kumar is Senior Staff Engineer Technical Standards at Qualcomm Technology Inc. At Qualcomm he is involved in consulting, research, and standardization efforts in the area of vehicle to everything communication security. He has had extensive involvement in the original design of the Security Credential

Management System and its subsequent security and privacy enhancements. He currently serves as chairperson of Security and Privacy Working Group at 5G Automotive Association, 5GAA. And has over 13 years of research experience in the area of cryptography and information security. It is my pleasure now to turn over the presentation to your first Speaker. Dr. William Whyte.

Dr. William Whyte: Thanks Nicola for the introduction. Welcome everyone to this course. I hope you find it useful. This course is about the SCMS, the Security Credential Management System, which is used to provision devices, vehicles, RSUs, and other participants in the V2X System with security credentials that allow their messages to be trusted by other system participants.

This course has five learning objectives. The first three we're going to deal with in this part, the same three we're going to deal with in the next part. The first three are we're going to define communication security requirements in the connected vehicle environment. And that learning objective is targeted at high-level decision makers and people who need a background in the issues that the SCMS addresses. So having been through that learning objective you should be in a position to understand the concepts and concerns that will get thrown around or a discussion of using the SCMS in a deployment. The next learning objective gives us a bit more technical detail at describing how the Security Credential Management System uses cryptographic building blocks to provide trust. And then the last learning objective in this part of the course will be given by my colleague Dr. Virendra Kumar. And it's going to start on understanding how to get devices interacting with the SCMS in a deployment.

The second part of this covers the last two learning objectives. Dr. Kumar will talk about the V2X certification process for a device to enroll in the SCMS. Finally, I'll take learning objective five and talk about how to make a deployment plan that uses SCMS services. So by the time you've got to the end of this course you should understand what you need to be thinking about if you're involved in a deployment to make sure that you can use the SCMS properly. One of the directions this course is trying to encourage you to go in, unless you're an actual SCMS provider already or work for an organization with experience in public key infrastructure, this course is going to encourage you not to build and operate an SCMS yourself. Instead to get SCMS services from a provider who knows what they're doing.

And so by the time we get to the end of this course you should understand how to make a deployment plan that use the SCMS. And you should understand what questions you should be asking the SCMS provider. At what stage in the process of deployment you need to be asking and answering those questions. And how integration with the SCMS will integrate with your deployment plans as a whole. So just reiterating Part 1, we're going to be looking at communication security requirements. We're going to be looking

at how the SCMS uses cryptographic building blocks to provide trust. And we'll be looking at how to understand how to get devices interacting with the SCMS in a deployment. Learning objective one communication security requirements. So the security of connected vehicle deployment depends on a large number of aspects of the system, and on those aspects of the system being properly secured. So support networks need to be secure against cyberattack. People shouldn't be able to hack into your systems. Stored data must be managed in an appropriate way with suitable access control. Each individual components or subsystems within the overall system must also be secure. In other words, you can have a secure system where individual firms have been properly secured, and obviously communication between components need to be secure as well. So those four items all need to be addressed.

The SCMS just saying we've got an SCMS doesn't address all of those items. It addresses the last two. SCMS helps with communication security because it provides devices but provides certificates that allow devices to communicate securely. And it helps with ensuring that individual components are secure because part of the SCMS's job is to issue certificates to devices only if they can be demonstrated to be secure. So having access to an SCMS is a vital part of a secure deployment, but other security issues must also be addressed for those deployments actually to be secure. And there's a companion course that's CSE 202 on cybersecurity that addresses those other topics, the topics that are raised by our first two bullet points above.

The purpose of the SCMS is to allow participants in the system to trust each other. If I'm driving a car and I get a SPaT message from an RSU, roadside unit, how can I be sure that that's genuinely generated by a real RSU? That it reflects the RSUs current state and so on, and so on. And this is difficult in electronic system because electronic communications can be intercepted, read, and altered. So obviously, this isn't a new problem. So there are existing techniques, secure communication mechanisms, which protect against this. And fundamentally those rely on cryptographic algorithms to meet the security goals for protecting the data in transit. So when you have data in transit, you have three top level goals: confidentiality, integrity, and authenticity. They address the three desires that the receiver has.

So I said confidentiality, authenticity, and integrity. But we'll address them in the opposite order, integrity means that the receiver knows that the message received is the same as the message that was sent. Or to be more specific, the receiver knows that if the message was changed they can tell that it was changed. They can't necessarily know what change was made, but knowing that it's different from the message that was sent is enough to let you know that you should be careful about crossing it. Authenticity the sender can demonstrate the receiver that they're legitimate, and part of legitimate mean it depends on setting. In this case like we were talking about with the RSU earlier the vehicle needs to know that this is the correct RSU. It's owned by the correct IOO. It's

in a good state, that it's communicating properly with the signal controller to get SPaT data and so on and so on. All that is covered by the authenticity.

And finally, confidentiality is used in some cases in V2X and not others. Confidentiality gives us assurance that the message hasn't been read by anyone who shouldn't read it. Now, obviously for basic messages, for SPaTs there is no such thing as somebody who shouldn't read it. There's broadcast messages. But if you're looking at more specialized uses such as, for example, towing, or if you're looking at vehicles re-provisioning themselves with additional certificates in that case there's going to be some personal or private information has changed, and so it's important that that can't be read by just anyone, so that's provided by confidentiality. And if we didn't have these three properties, especially the first two integrity and authenticity, there wouldn't be any point in sending messages.

Receivers couldn't be sure the messages were correct. And the system could easily be flooded with false messages because there'd be no way of stopping bad senders from sending them. Receivers won't be able to act on received messages because they could so easily be forged that there wouldn't be any point to act on them. In any communication system including connected vehicle trust in the received messages is vital for the system to achieve its goals. So as I said, we already have mechanisms that address this are well known. Almost everyone I'm sure has heard of SSL or TLS, which is the protocol used to protect to secure web browsing. So what's special about connected vehicle? Why do we need a whole course about SCMS? Why do that course need to be this size?

So there's some significant differences between connected vehicle and the trusted communication you might be used to in the context of the web. In the context of the web if I go to ecommercegiant.com my browser makes sure that the server is connecting to is actually owned by that ecommerce company. It does that using TLS and X.509 certificates. And X.509 certificates are these digital documents using cryptography that show that someone has checked that this website belongs to this company. Or more generally X.509 certificates assert identity. In the context of the web that's all I want. If I go to ecommerce giant.com versus eauctiongiant.com here, I need to make sure that I'm connecting to the website that I intend to connect to, but that's all the information I need.

In the CV system we've got more complicated requirements. First of all, we don't have this hub-and-spoke topology. We've got many-to-many communications, peer-to-peer communications. Second, we need to ensure the receiver can make real-time decisions. So all of the information you need to make a decision about is this message trustworthy you need to receive with the message or know in advance of it. You can't assume you're online. You can't assume you can go off to some server and get additional

information to decide whether or not to trust because that won't have a latency that matches with safety of life situations. And it's true that in the future a large number of devices will have good low-latency network connectivity. but that can't be assured for all devices. And so the system has to accommodate those devices that can only get a home line from time-to-time. Another thing about CV system requirements is we're looking at roles not at identity when it comes to making security decisions. So if I get a message from *your* car that it's *a* car and I don't care if it's your car. And if I got something that said—that has your name in it, that wouldn't be any use to me. Just because I know your name doesn't mean that I know you're entitled to send basic safety messages.

We've got this very different set of attributes of the standard that we want to prove in the connected vehicle system, compared to online. In the web saying you need to prove identity in the connected vehicle system. You need to prove your ability to act in a role. Also in CV because it's managed many we have concerns about privacy, about you giving away your routes about you being trackable. That you don't have in the web where it's one—where it's many-to-one. And there's one sector where your data ends up. And, obviously, that center can potentially breach your data, but you don't have to be worried about your privacy in the context of the communication session. You just have to be worried about it in the context of what might happen afterwards. While in V2X, you need to be concerned about the privacy implications of all of these messages that you might be broadcasting.

And finally, this is the same underlying constraint that drove our need to be able to make real-time decisions requirement involved, but specifically in the case of security you've limited connectivity for security update. So if I need to get new certificates, yeah, that provisioning process needs to work in a context where I'm not guaranteeing connectivity and so it needs to be slightly opportunistic and ad hoc. As far as many-to-many communications goes here we see a nice illustration of it. All of these cars are talking to each other. The roadside units are talking to all the cars. You see all these information that's getting distributed to the cars, speed limits, warnings, road conditions. All of this is many-to-many. And so that just illustrates that we need to have this kind of role-based authentication where when you got a message you're told what the sender is allowed to send so you can make that decision on the spot.

Again, the need for local real-time decision for what we have here on the left is cars talking to each other. What we have on the right is a picture of the SCMS, and all the information about who's good and who's bad, who's trustworthy and who's not is stored inside the SCMS. But these cars on the road don't have the time to go back, and get that information from the SCMS when they're making a safety of life decision. So the system needs to work here. We have this X here between the two pictures. The system needs to work in a context where the devices in the field can't go back to the SCMS with

every message and go is this one okay? Is this one okay? Identify versus role, again, we've talked about this a little. A pedestrian may want to be able to request a pedestrian crossing signal. A pedestrian may want to be informed of a pedestrian crossing opportunity. Ordinary vehicles can send basic safety messages. Police cars can send signal preemption.

And in all of these cases you see there's a category of actors, people, or vehicles that can send particular types of messages. An RSU is the only who should be able to send SPaTs, for example. And so inside the system we're interested in can we say for certain this device is an RSU? Can we say for certain this device is an ordinary BSM sending vehicle? Can I say for certain this device is a police car? Again, we have these concerns about privacy. We're sharing a lot of information about ourselves, position, brake status, speed, acceleration, that kind of thing. But the system needs to work without me giving away in my message an identity for myself or an identify for the vehicle like a license plate number for the vehicle.

The purpose of the system is to improve safety and mobility. And the system works best if everybody who can use it does. And so it's not just the right thing to do ethically to make sure the position preserves privacy. It's also the right thing to do practically. The less of a threat the system poses to privacy, the less of a risk there is that people would choose not to participate. Again, limited connectivity for security updates for vehicles.

Here we've got a map of the Detroit area. And there have been times in the past when there'd be two access points to the internet for vehicles driving around. And if a vehicle is near one of these access points, it's simply not going to be able to get an update. So the system needs to be rebuffed in the case where vehicles go a long time, potentially months on end, without getting access to the open internet to update their security management material. How did this translate into what the system does, our functional requirements? We're going to issue devices with credentials. They're called digital certificate in this context. And those credentials state the properties and permissions of the device. They say everything the receiver needs to know about the message to decide whether to trust its message.

We use cryptography in this context to make sure that only the owner of the credential can use it. So we're tying the credential to a private key. We'll talk more about what that means at the next learning objective. Until only that private key is protected with hardware so they can't be extracted with software so that you can't have malware asking you to private key without being entitled to. Then only the sender of the message, only the owner of the certificate can use it to sign those messages. So the credential issuer can't sign messages. The credential receiver can't sign messages with it. Only a legitimate sender can sign messages with it. The issuer should always make sure the senders are entitled to the credential they're asking for. And finally, we want to

have a requirement of the system you can recover from compromise. So if there's misbehaving devices that send bad data or otherwise from the system they can be detected and removed for fakes. If we have bad actor on the system management side like bad credential issuers they can be detected, and credentials they issued can be removed from circulation.

So these are how we're going to go about working on our system design. So these properties requirements, which we see here in gray on the left they drove the development of a different certificate format than we're using on the internet. On the internet we're using X.509 certificates as I mentioned before which are really suited for identity management. In this context, we're going to be using 1609.2 certificates.

We also have limited capacity channel. So there's limits on the amount of data that you can squeeze through channels. Obviously, at the moment in the United States, C-V2X is the radio access technology of choice. Previously, we worked with DSRC, and the system was designed with DSRC in mind. That point there is six megabit per second channels, which could easily be saturated if too many people were present. And even with 20 megahertz channels in C-V2X those channels have higher capacity, but they can still become saturated, especially if the security overhead is excessive. So one reason for having a new certificate format was to reduce the security overhead. A small certificate format was necessary to do that, and 1609.2 is smaller by design than X.509.

Another thing is to support this role-based access control that we've talked about where 1609.2 certificates are designed to identify certificate holders by their role and permissions. The role that you're playing is identified by the provider service identifier or PSID. And that's an identifier that's managed by IEEE to make sure that each use of a given PSID refers to the same thing. To be more concrete about that if I'm sending basic safety messages I use PSID hex 20. If I'm sending SPaTs I use a different PSID. And that's kind of the fundamental identifier in my certificate of what I'm allowed to do. Then even within applications we can have some more granularity.

At the high level, think about PSIDs as sandboxing the system into different areas, different sets of activities that different participants can do. So 1609.2 certificates use PSIDs, other kind of fundamental unit of identifying the properties of the device, the holds, the certificate, X.509 certificates naturally identify participants by identity. And that's not so useful in the 1609 system or in the V2X system. So 1609.2 certificates meet the system requirements. And given that we have these new certificate formats, we have new certificate management protocols compared to X.509. X.509 is mature and well understood. But if you're moving away from the certificate format, you can't use the certificate management and protocols. They've been built to handle that certificate format until we ended up needing to design the entire SCMS as well. So those are our security functional requirements.

Then let's talk about privacy requirements. Privacy basically means you have a right to go about your business without all the people knowing what you're doing unless they need to. What does that mean? If they need to know what you're doing to provide a service then they can learn what you're doing. If there's a public safety reason for them to know what you're doing. If you give your active consent, or otherwise depending on local privacy regulations. You have a general right kind of aspirational right to going about your business. But even when that waters down, others shouldn't get more information than they need here where the information that they need is defined in the context of these exactly, they—providing you a service. They need the information that is needed to provide the service. You do have a right to go about your business, but the V2X system inherently introduces risks. It potentially leaks information about your movements. You can be identified with a vehicle.

One observation of your vehicle sending a basic safety message potentially allows somebody to track you by seeing basic safety messages that you send later on down the road a mile or five or 50 miles. And one of the things we'll talk about is how we've been very careful in the design to avoid that happening. So in principle an observer could see you in a number of different places, different times, and recognize that all of those transmission have come from the same vehicle. In practice, we've made that extremely difficult. So the design goal has been to ensure that V2X communications are never the cheapest way to track you. And we put it like that because there's lots of ways you can be tracked in real life. You have your license plate. You have your toll tags. You have your phone in your car. Shops track your visits. The tire pressure sensors in your car give out low-power unique identifiers. They're hard to hear from far away, but they can be heard by somebody outside the car. And in general your car is now full of radios, those radios are MAC addresses. A MAC address, readers can potentially read them well, the point of V2X is it's trying to be longer range than a lot of the wireless traces in your car. So obviously it creates potentially an increased privacy risk.

So the appropriate goal is to make the V2X not the cheapest way to track you in this world where all the ways of tracking you exist. There's no point having a different goal because if you go beyond not being the cheapest way you're not improving privacy. The attacker will simply use the cheapest way available to them. So that's privacy requirements on the send side. There's also privacy requirements on the receive side. Your vehicle is generating a huge amount of data, 10 BSMs per second. It works out the nine megabytes per car. In an hour's driving it works out to 5.4 by 10^{15} bytes generated by all the cars in the U.S. So some agencies and operators and organizations have legitimate interest in that data so they can do traffic planning. And may want to gather that data and make use of it. So we need to ensure that the data is appropriately managed and gathered only when necessary and managed in accordance with data privacy and data management principles. And it's deleted when it's no longer necessary. It's anonymized before it's made available to others and so and so on.

This data management receive side as you probably noticed by now the SCMS is really about stating properties as sender of a message. Data management isn't really an SCMS message issue. We're just mentioning it here because it needs to be addressed for deployment. So how does the SCMS address all of this? SCMS uses 1609.2 certificates. Those in turn specify security services, cryptography, and data validation services. They can be used to protect data in transit. In that 1609.2 system the receiver knows of sender's trusted to send a message or command of particular type because the sender has a certificate that says they're entitled to send that message. And the 1609.2 processing cryptographically links the certificate to the message, and that shows that only the certificate holder could have generated the message.

So basically, if I send a message with particular contents the certificate says I'm entitled to the same content. And then the cryptographic linkage the digital signature shows that I generated the message. So that it's kind of triangle information. The SCMS is in charge of issuing certificates to actors in the system. And its primary responsibility is to make sure certificates are issued to actors who are entitled to it. That means the SCMS is responsible for carrying out checks or making sure the checks have been carried out to ensure the first the actor was entitled to the certificate in the first place. You're not giving a signal preemption certificate to an ordinary car and saying the actor has become malicious or untrustworthy or otherwise unreliable since the certificate was issue.

So the SCMS need to have mechanisms in place for what's called misbehavior detection and revocation so that if devices start to be non-trustworthy it turns out that they're sending out bad information the SCMS can take action to ensure that receivers don't listen to that device anymore. The major challenges we're going to talk are enrollment provisioning and revocation. Enrollment is this thing we've talked on a number of times already about how it's the SCMS's job to make sure that you're entitled to those certificates. How exactly does the SCMS do that is going to depend on the specific application. The way you demonstrate you're entitled to be a police car is different from provisioning of ordinary OBEs, which can probably be done at the factory. So each of these need to be designed on an application-by-application basis. That's why there's a slight challenge in deployment. The more applications you have, the greater complexity around enrollment.

Provisioning is keeping devices provisioned with certificates. This requires you regular access to the Internet. And this needs to be done in the context we talked about where actors don't always have good access to the Internet. They may have intermittent access. So how far in advance are they provisioned with certificates? If you've been provisioned with certificate in advance, then, obviously, if you get compromised or misbehaving that allows you to continue misbehaving until the certificate's thrown out. So there's a balance, and the SCMS and deployers need to understand how to strike

that balance between ability to keep broadcasting if there's an outage versus vulnerability to bad devices if you can keep broadcasting for too long. And so that's linked in with our final challenge, which is revocation, understanding what devices needs to have done to be denied the opportunity to broadcast anymore. So, for example, if I am a vehicle and I'm sending out basic safety messages that every so often just arbitrarily show me hard braking even though the vehicle itself isn't hard braking that would probably be something that would need to be revoked. That's going to be causing all sorts of alerts and confusion. It's going to be causing hard braking events behind those.

And so revocation is a way of telling receivers no don't cross message signed with these certs anymore. But again understanding who gets revoked, what they get revoked for, and how the revocation information gets distributed is still a significant challenge in SCMS deployment. So that's our background.

As a check on our learning let's just look at this question. Which of the following statements about privacy is not true? There are many ways to track drivers on the road. Or to preserve privacy consideration must be given to how data is created, transmitted, stored, and managed. Or protecting privacy both technological and policy approaches. Or the V2X system must always completely protect the anonymity of drivers. So which of these statements about privacy is not true?

You can pause this. So I'm going to go straight on to the answer. So the correct answer is the V2X system must always completely protect the anonymity of the driver. The correct answer in other words, the thing that isn't true. So the driver privacy is important, but it can't be fully guaranteed by V2X. It's somewhat compromised by many other mechanisms, and so the V2X system design aims to ensure the V2X isn't the cheapest method available to an attacker to compromise privacy. The incorrect answer is first, there are many ways to track driver on the road. That's incorrect drivers—sorry. Incorrect as in it is true. I'm sorry if I put negative phrasing of the question. So there are many ways to track drivers on the road is a correct statement. So it's an incorrect answer. Drivers can be tracked in lots of different ways. To preserve privacy consideration must be given to how data is created, transmitted, stored, and managed is a correct statement, so an incorrect answer. Data may be personal identifiable information and must be protected. Protecting privacy uses both technological policy approaches again is a correct statement, so an incorrect answer. So the correct answer, the incorrect statement, is answer D. So that's learning objective one. Thanks for your attention. And hopefully even if you can't sit through the rest of the course, hopefully, it'll have given you some understanding of the issues that we're grappling with the SCMS. And how the SCMS design begins to address those.

In this next learning objective we'll dig in, in a bit more detail, talk about how the SCMS uses cryptographic building blocks to provide trust. Excuse me. So as we mentioned, trust is essential to this connector vehicle system. If receivers can't trust the message, they're useless. And so digital signatures are the cryptographic building block that we use to construct systems of trust. In a digital signature system, senders sign messages with digital signatures and receivers verify the digital signatures. To verify the message you need to have a public key and that's contained in this certificate that we've been talking about. The certificate is sent with every message or periodically. The certificate is issued by the SCMS, and that's where you get both point of control where policy is applied or the SCMS determines that a certificate requester is entitled to get the certificate they're asking for. And message signers are called end entities. So we have two broad types of components inside a public key infrastructure the CA, the Certificate Authority, the agency certificate. And the end entity the use of the certificate. And if I'm an end entity with a certificate that shows first I've correctly implemented the application. I'm rolling it on a properly secure device. And I'm allowed to send a message of that type. Sorry. Again, you could have a correctly amended police car application to do single preemption. It could be wrongly on a properly secure device. But if that device is not actually owned by a police department, it doesn't matter that it's secure enough. It's still not entitled to the certificate.

So this mechanism is the digital signature, and it provides communication security services or authenticity ensuring the message came from a stated sender. Integrity showing the message modified on the way and non-repudiation, which means that the message sender can't deny creating the signature. What that means is that if I receive a message signed by you I can take that to some third party later on, and they can be satisfied that you signed the message. There's no way you can say oh, no, I signed—you know, the other person signed the message. The non-repudiation is very useful in a setting where messages may be used as evidence later.

How do digital signatures work? First of all, the sender before sending any messages they generate a pair of cryptographic keys the public key and the private key. You'll typically generate the private key first and derive the publicly key from it. And they're mathematically linked so that if anybody knows the private key they can get the public key. But if they know the public key, it's very, very hard for them to get the private key. So the public key is linked to the private key by some kind of mathematical operation. But the larger the key is the heart of the operation is carried out, and it's exponentially more difficult so even quite short keys give you a very high level of security. In a connected vehicle we use an algorithm called ECDSA, the elliptic curve digital signature algorithm. And that's standardized. It's used by NIST. It's used across Europe. It's a well-known, well-understood cryptographic algorithm with very nice performance properties. That's signatures, of course, the other service we talked about is confidentiality, which means ensuring a message can be read by an outsider. And

encryption, which I'm sure most people are familiar with, encryption is the operation or the mechanism that gives you confidentiality, that gives you that assurance.

There [are] symmetric and asymmetric encryption key schemes. Asymmetric schemes are like the public key crypto we talk about for signatures. You generate a private key and a public key, and then you distribute the public key widely. Anyone could get the public key, can encrypt with it. But only you who owns the private key can decrypt with it. And then the other approach you have is symmetric where there's a shared symmetric key. And anybody who knows the symmetric key can both encrypt and decrypt. Those, obviously, differences between the two. And asymmetric seems in here to be more secure, which in a sense it is but symmetric crypto tends to be much more fast and involve much less overhead. So in connected vehicle and a lot of other settings we use hybrid encryption where asymmetric algorithms are used to establish a symmetric key, and then the efficient symmetric algorithm is used to encrypt the data. I should also note that as we mentioned earlier, conventionality isn't as big a deal in connected vehicle as it in some other settings. In most V2X messages aren't encrypted with BSMs, MAPs, SPaTs. They are used for things like protection of financial information, or secret data exchange between end entities and the SCMS. So in this presentation we're going to be focusing on the asymmetric cryptography that enables the usual digital signatures and asymmetric encryption. Those in turn enable secure ad hoc networking.

So we've already talked about how the SCMS's job is to make sure that a certificate requester is entitled to the certificates its requesting. The specific component inside the SCMS that does it the Certificate Authority or CA. So it allows us to establish trust between two previously unknown users. CAs have a private and public key of their own. And they use the private key to sign the certificates. And then anyone can use the public key to verify the certificates, so we have this chain of trust starting with the end entity and going up through the CA certificate where everyone's permissions are established by the certificate. And each certificate can be verified using the public key of the next entity up in the chain.

CAs are trusted to keep your private key secure. And they're trusted to ensure that the users are entitled to that specific certificate. And in a lot of models we separate those two out. We do that in V2X as well. There's one part of the certificate issuance system is the, RA, the Registration Authority. That checks that you're entitled to the certificate. And the other part is the CA, and that actually does the mechanics of issuing the certificate. So we mentioned the chain of trust that Bob here is a car. He gets his certificate from a CA. The receiver needs to trust Bob's certificate. They haven't seen it before, but they can trust Bob's CA certificate, then the use the CA, its public key to verify Bob's certificate. Use Bob's public key to verify Bob's message. And if the

receiver doesn't already know Bob's CA certificate, we have this chain going all the way back to what's called a Root CA that's a standalone CA that issued its own certificates.

And there's some complexities around the management of Root CA certificates, which we'll talk about later. The important thing is that in any kind of civilized system there's not going to be a large number of Root CA certificates. So managing them becomes much easier than manually managing all of the end entity certificates would be. So Bob gets a certificate from the CA. Alice receives the message and needs to already know Bob's certificate, or already know Bob's CA certificate, or already know one of the other certificates in the chain. And so we're going to assume that you get provisioned at the start of your lifetime with all of the certificates for the Root CA. As long as you got that, you can then trust any incoming message. The 1609.2 certificates that we mentioned they're a smaller size. They're suitable for machine-to-machine communications. They allow for pseudonym certificates meaning that they can conceal the identity of the owner of the certificate. And they also allow the SCMS to issue certificates for all types of CV application using PSID. And the fact that we've got the PSID which is this identifier matched by IEEE means we can add more applications in the future, so drones, autonomous delivery robot identification, different flavors of tolling.

Any application that gets defined in the future can have certificates defined for an application. The next few slides we're going to dive in a bit more detail, so now we're going to do a bit of a deeper dive into it a SCMS design, not just its property. So at the core the SCMS is a standard public key infrastructure, Root CAs, intermediate CAs and end users and authorization CAs. And we see that standard PKI chain of trust running up the middle. We've shown one intermediate CA. There can be none. So authorization CAs can get certs directly from Root CAs. There can be one. There can be more than one. There's different conventions in different regions. A participating device in the system needs first of all its own authorization CA to get certs from. And second, a collection of certs for all of the different Root CAs, so they can trust incoming messages no matter which Root CA ultimately issued the cert that's being used to authenticate that message.

As we mentioned, root cert management is complicated. This functional box up here the electors is responsible for managing the Root CA certs. SCMS has a lot of different components. And in principle each component can be run by a different company. The design is modular. Internal interfaces are documented, although not formally standardized. So you could have a Root CA run by Alice's SCMS services. An immediate CA run by Bob the SCMS services. Authorization CA run SCMS or Alice and so on. In this case, as deployer you engage with the Registration Authority operator, and they'll interface with the rest of the SCMS operator. But in practice although you could have this modular deployment SCMS functionality in practice there are different SCMS providers who run everything from the Root CA down. Because of this Root CA

management function that we talked about, if you have different deployers, if City A works with SCMS Provider A, City B works with SCMS Provider B, you can still trust messages. You can still drive a car between the cities and trust all the messages because of this sharing Root CA information.

So this is in practice the deployment organization that you'll most often been working with. And so as somebody in charge of a deployment there's just one SCMS provider you talk with no hidden complexity organizationally behind the scenes. There's two different types of certificates as well. We've talked about authorization certificates, which is the one to use when you're sending application messages. You also have a long term what's called an enrollment certificate that's used for interacting with the SCMS. The enrollment certificate, it's used for certificate management interactions that authorizes certificate requests, and download for authorization certificates. And it's essentially a key used by the SCMS to look up your metadata, such as what permissions can go in the authorization to get this. We've separated these because that ends up improving robustness and creates a safe privacy within the system as a whole. End entities will directly contact only three components of the SCMS. Initially, you can write your enrollment CA, your enrollment cert. Then when you're out in the field, you will contact the Registration Authority. That's the gateway for SCMS interactions while you're in the field.

You in this case [are] the system participant like a vehicle, not a deployment site. The RA helps you provision yourself with certificates. It helps with your misbehavior report upload. We'll talk about that later. It lets you receive new certificate revocation lists, new Root CA information. And so that's basically the go-to for any security management information you need. For robustness, the system might also provide a distribution center that's an alternative path to download security management information, to download public security management information.

So CRLs, new Root CA information, and so on and so on. But if you're getting new certificates, topping up with certificates, that's always done through the RA. It's never done through the distribution center (DC). The points of contact with the SCMS again enrollment CA, RA distribution center. And then eventually contact the enrollment CA at the start of its lifetime, and that can be a proprietary interface. There's a standardized interface in IEEE 1609.2.1 but it's not required. Then as part of initialization each device gets hardwired with a URL for the RA that it's going to use. And the device will use DNS to map that URL to an IP address, and then go to the IP address and establish a secure connection with the RA. As far as the distribution center goes, again, a device will be configured with URL through distribution centers. They may be distributed over the air. They may be managed through some software configuration management process. And the senders then talk about how the end entity is configured with the distribution center URL.

So if you're involved in a deployment that wants to use distribution center for robustness then you need to with your suppliers to understand how their devices can tell what distribution centers use. These interfaces are all standardized in IEEE 1609.2.1 which is published in December 2020. There was a previous interface called the CAMP interface. That's widely deployed as of the end of September 2020, but deployments are migrating 1609.2.1 And at the time of presentation of course U.S. supporting roles were migrating from DSRC to C-V2X. And all of these migrations can happen in conjunction with each other.

We talked a little about privacy how we want to prevent eavesdroppers from linking messages that you sent in different locations. And one way we do that is with a special type of organization certificate called a pseudonym certificate. So pseudonym certificates don't contain any identifying information about the sender, not the driver license number, not the license plate number, not the VIN. They simply say you're entitled to the BSMs. And the BSM sender has multiple pseudonym certificates. They're all valid at the same time. And so that means the sender can choose for each message which certificate they want to sign it with. And in practice you won't use one certificate for one message, a different one for the next and so on, and so on because anyone who's near you at the time of the change can work out that those certificates come from you. So you actually get better privacy by changing the certificate occasionally every five minutes or so. If you do that, then it's only over here you have Point A. And then over here you have Point B, ten minutes later. You'll be using different certificates. You'll be using different other identifiers inside methods and so it becomes very hard for them to tell well were these two observations in sync? Or were they two observations of similar but different cars.

So you do that with your collection of pseudonym certificates. You have 20 to 60 pseudonym certificates a week. It varies from place to place. The convention in the US has been to have 20 a week. In Europe, they're appreciating more towards 60 to 100. But by changing your pseudonym certificate at a strategic time you've made it very difficult for an eavesdropper to track you. We'll talk later about how you might do revocation in a way that's consistent with the privacy property of pseudonym certificates. We have material on that later on. There's one clever feature about the SCMS design called butterfly keys, where in order to manage network traffic better and manage peak load demand on the SCMS, and end entity instead of standing up a separate request for each certificate it needs just sends up a single request at the start of its lifetime. And then this butterfly key mechanism is used to generate different key pairs for all of the certificates that it's going to need.

You can generate three years' worth, 10 years' worth. And 100 percent secure in that nobody can look at the butterfly key output and tell the two keys were generated from the same seed. And nobody other than the original requester can sign with those keys

because there's one secret part of the seed that never leaves the requesting device. So this single request then generates all 20 or more certificates for a given week, and all the certificates for all the future weeks. That gives you greater efficiency managing when certs are going to be generated and better privacy because it decouples the time of generation from the time of request.

So the Authorization CA can't link different requests together that were received at the same time. One thing to note, of course, is the—as we said there are lots of applications in the system that an RSU might send, WSAs might send SPaTs. And so there's a question about does the RSU have one certificate that it uses for both of those operations? Does it have one certificate for WSA, one certificate for SPaT? There's a trade-off here storage space versus organization complexity. Right now, there isn't a really strong convention. And your SCMS provider may have policy that covers which application share certificates. It may say WSA and SPaT needs to be in different certificates. It may say WSA and SPaT needs to be in the same certificate. It may see we can support either. How do you want to do it? In principle, the SCMS could also say if you have a device that's doing Application A it can also do Application B. There's not really a convention of the SCMS making that kind of policy either. The SCMS has tended to make policy about what goes in certificates more than what goes on devices. But if it were to choose to make that policy, it could.

So again, it's just being flagged in case the situation comes up. Right [at] the moment it doesn't. And I think the baseline is unless there's a really good reason like a really strict space constraint that needs you to want to minimize the number of certificates. Unless there's a really good reason to couple two applications together in the same certificate it's probably easier to have separate certificates for separate applications.

So now let's talk about misbehavior management. We've talked about the PKI. We've talked about end entities interactions. And then what do we do about bad end entities? The system allows for devices that send bad data to be removed. Bad data means it's not real. It's describing a situation that don't exist. So if a car says in midair that's bad data. If two cars say they're in the same place at the same time that's badly. If a car says it's braking but then speeds up, that's bad data. We've defined these mechanisms and end entities cars or RSUs can use, so that if they see this bad data, this misbehavior, they can report misbehavior to their RA. And the RA passes the reports to a central Misbehavior Authority, the MA. And the MA analyze[s] the reports and decide[s] whether the bad data that's being reported is so bad that you need to do something about it. If it's really bad, then the MA says, okay, we need to revoke that device. And for design reasons and crypto security reasons, those aren't worth going into in detail, you need to be revoked by a CA in the chain of trust. The MA itself doesn't issue the revocation. Your CA if you're being revoked issues that revocation. And your RA is told okay this device is being revoked. Don't allow it to get anymore authorization

certificates. So the process of informing the RA blocked you off on the supply side of getting certificates.

The process of issuing the CRL protects other devices on the receive side. So they get the CRL that says this certificate is no longer to be trusted. And so if they receive a message signed with that certificate they don't trust us. Here's an illustration of how the CRL works. So the CRL is what's used by receivers to protect them against bad senders. CRL, certificate revocation list, is the list of revoked certificates. So revoked means that you're being told not to trust them anymore. And so if I'm a receiver I get a signed message. I check to see has that been revoked. In other word, is that certificate on the CRL? If it's been revoked then the message is invalid because the revoked certs can't be trusted. And we've two ways of managing bad certificates via revocation. One is you include them on the CRL. The other is you simply let them expire. If a certificate has expired, in other words, if the current time is after the expiry date in the cert, then you don't trust the cert anyway. So that makes management simpler. It also keeps down the CRL side.

Once a certificate has expired it doesn't need to be on the CRL anymore because it simply isn't trusted. So the way we organize the CRL is that each certificate can only be on a single CRL Series. So each CRL signer can only revoke a particular set of devices. Each device can only be revoked by one CRL signer. And this is the concern I was talking about earlier where we've kind of locked down exactly who can revoke certificates in order to reduce the damage that could be done if there was a rogue CRL signer in the system. If a CRL signer goes rogue, they can revoke those devices that they're responsible for, but they can't revoke absolutely everyone. So in each certificate there's a CRL ID. When I receive a message that's signed by a certificate. I look at the CRL ID in the certificate. That tells me which CRL I can look at, the one with the same CRL ID. And if there's an entry in that CRL that matches the certificates that means the certificate is revoked.

For pseudonym certificates it's a bit more complicated. We've got 20 or more certificates a week. We still have the same CRL ID in the certificates. But these certificates are revoked using a linkage value that appeared on the certificate, which allows a single CRL entry to efficiently revoke all of the vehicle pseudonym certificates. So now what happens is I get the CRL ID from the receive certificate. That lets me look up the CRL that may be used. And now each of the entries in the CRL is what's called a linkage seed and that linkage seed expands into a large number of linkage values. So I pre-process the CRL. I've checked. I've generated the entire set of linkage values. And now if any one linkage seed, it produces a linkage value that matches the linkage value in my certificates, that means the certificate can't be trusted. Now, the way it's to help each of these other linkage values that were generated by the same linkage seed will correspond to all of the other certificates that I have. So this one revocation entry

revokes all of the certificates that I have. But the clever thing about this from the point of view of privacy is that if the linkage seed isn't published there's no way that you can look at two different linkage values. They disperse from the second one. There's no way you can look at them and know that they've been driving the same linkage seed. So this is perfectly privacy-preserving until you've been revoked. It's also privacy-preserving back into the past.

So if my certificate is revoked at Time T by a CRL you know all of the certificates that belong to me at Time T, but that doesn't help you work out the certificates that belong to me at Time T Minus 1, or any other time in the past. So revocation is our mechanism for removing misbehaving devices. And as I mentioned preserving privacy is important at the system. And so the SCMS design ends up being fairly complicated in order to allow us to do misbehavior investigation and revocation in a way that preserves the privacy of innocent participants in the system. So, the information that you use to determine who owned a pseudonym certificate is distributed across the system through these four components, the ACA, the authorization CA, the CRL signer, the Linkage Authority the LA1 and LA2, and the RA. The linkage authorities are involved in generating the linkage values. And so, the ACA, the LA, and the RA all need to collaborate in order for the replication to happen. And they collaborate through well-defined interfaces, but make it hard for anybody to go on a phishing expedition and learn things about the behavior of vehicles that they shouldn't unless that vehicles been misbehaving. As you can see it's a complicated system, and there's still some details to be worked out.

So the mechanics of generating CRLs, the mechanics of generating search that can be revoked with those CRLs, those are all well understood. And what that means is that say somebody has a car, and an attacker gets over the car, extracts the private key and posts it on the Internet. If an attacker does that we can still revoke the car. We don't need any misbehavior reporting. We can just put the car on a CRL. But for misbehavior in the field, we're still working on defining it. There's work happening at ETSI. There’s work happening at in SAE. There’s preliminary work happening in the connected vehicle pilot deployments. More advanced research being done by the University of Michigan. It will probably be end[ing in] 2021 before there’s standardized misbehavior techniques. So if you're watching this training course at the end of 2021 then maybe go and see if there's another training course on doing misbehavior detection. But before then it's unlikely that there's going to be anything. And so the best thing for a deployer is to work with the SCMS providers to understand how they'll manage edge cases where misbehavior happens.

So those are all the functional components of the SCMS. At the top of the SCMS, we have the SCMS manager that sets the policy that those functional components have to follow. So the SCMS manager sets the policies. It audits the Root CAs to make sure that they're following the policies. It coordinates the approval of new Root CAs or

removal of roots CAs if they aren't following the policy properly. In Europe, DG GROW is running a certificate policy group that is playing exactly this role, the SCMS manager role. As of 2020, a U.S. SCMS manager has been established as an industry organization with observer participating from USDOT, but it's slightly ad hoc. It could be that in 2021 or afterwards it becomes more formally accepted. And so for deployment in the 2021/2022 time phase, it's best again to select just an SCMS provider. And understand from them who's in charge of managing their policy, who's in charge of ordering their CAs, so you can make sure that they're following best practices. We'll talk about this more in learning objective three.

So, as we've mentioned earlier one of the jobs of the SCMS manager is to manage Root CAs because there could be multiple different Root CAs, essentially because different Root CAs might be run by different stakeholder organizations, might have different priorities. And if you are from one set of stakeholders that isn't running a Root CA you might feel you weren't getting the level of attention, the quality of services, speed of turnaround that's necessary for your application. And so we wanted to design the system so that it's possible to set up new Root CAs, and have them accepted so long as they're following all the policies and all the other requirement. So, at the time of recording this presentation in late 2020, there's only one Root CA that's widely used in the U.S. but there are multiple Root CA suppliers. And in fact, in PlugFest in Europe, there have been up to 12 Root CA suppliers represented. So there's a wide choice of implementers to work with. If you are on your deployment you need to understand is support for multiple Root CAs required. So, for example, if I'm running a city site and there's going to be vehicles driving into the city, then I want to make sure that any Root CAs that vehicle certs chain back to will be accepted by my RSUs. So right now it's got to be done on a deployment-by-deployment basis, deployment managers decide which Root CAs they're going to trust in consultation with their SCMS provider. And they let their known supplier installs the appropriate Root CA certs on their devices. But in future there's going to be a system called the electors where we have three or more electors whose job is to sign a certificate trust list that lists all the trusted Root CAs. That's standardized in 1609.2.1. Once that gets widely deployed, it's going to make Root CA management significantly easier.

The electors are the last entity we need to talk about. They are the entities that sign certificate trust lists which are the list of what Root CAs are to be trusted. But the SCMS manager actually decides which Root CAs are trusted. The electors just sign the list. And this is a separation of technical responsibilities from policy responsibilities, basically so that we can have a single point of decision for the policy decisions. And multiple redundant systems for making sure that that policy decision is authenticated and properly communicated. So that protects us against the case where one of the elector's work get compromised. We don't have a single point of technical failure. And so if you

look at the processing an end entity gets in a certificate trust list that's been signed by the electors. The end entity trusts if the elector signature is verify.

So from a processing point of view it looks as if the end entity is trusting the electors. But from a logical and organizational point of view, the end entity is actually trusting the SCMS manager, and trusting the electors to do what the SCMS manager tells us. We have this quorum system where if a CTL is signed by, for example, three of five electors it's trusted. And that means that you're going to have failures of electors. You can remove electors. And it doesn't cause the system to break down. We have this clean transition mechanism even as electors age out and their certificates expire, you can start off with five electors, operate with four for a while, move back to operating with five. And it's all seamless from the point of view of end entities. So once that elector system is rolled out deployment managers will no longer need to take responsibility for specifying which Root CAs to trust. It will all be done through the SCMS manager. And currently SCMS managers already has stood up electors. There may be other SCMS manager organizations in future, but devices will be affiliated with a single SCMS manager that will help them through the process. The client software support for that mechanism is under development as of the end of 2020.

Okay. There is a lot of information there. Obviously, it's hard to have one question that captures the whole thing. If you look at the student supplement, you'll find there's a number of different questions about this learning objective and others. But for purposes of this presentation we'll just talk through this one. Which of the following is good practice for a CA? So the good news here is that the correct answer is the correct answer. It's not the incorrect answer is the correct answer. Which of the following is a good practice for a CA? First, share its private key with the authorities to assist with legal investigations of hackers. Second issue certificates to anyone who pays a fee. Third, require end entities to submit a copy of their private key before they receive the certificate. And fourth, ensure that certificate requesters meet minimum standards for the security and are entitled to the requested certificate.

So if you need more time to think you can pause. I will go onto the answers. And the last one is the correct answer. CAs are supposed to ensure that certificate requesters meet minimum standards for security, and are entitled to the request of the certificate. So let's look at the wrong answers. A CA shouldn't share its private key. If it shares its private key that lets anyone issue certs on behalf of that CA. Obviously, CAs should be prepared to cooperate with authorities under local regulations. It should never be necessary to share a private key. Should CAs issue a certificate to anyone who pays a fee? No. The mere fact that you paid a fee doesn't demonstrate you're entitled to the certificate you're asking for. The CA needs to ensure that you're entitled to that certificate. Should the CA require end entities to submit a copy of their private key? No for a similar reason why answer A is wrong. If the CA knew the end entity's private key it

could generate messages, and make it look as if the end entity had generated them. So that would give a rogue CA immense ability to create messages that cause confusion in the system. So again, the CA needs to make sure the end entity is entitled to the certificate, but it doesn't need to see the end entities private key to do that. And that's the end of learning objective two. Thank you for your attention and I'll hand over to Dr. Virendra Kumar.

Dr. Virendra Kumar: So in this learning objective our goal is to understand how to get devices interacting with the SCMS in a deployment. So in a deployment there are four types of parties interacting and participating in the system. The deployment manager engages with the SCMS provider and the device supplier. The SCMS provider creates requirements—policy requirements that a device supplier needs to fulfill. The device supplier, on the other hand satisfies those requirements provided by the SCMS provider. And it uses that to get certificates from the SCMS provider. The device supplier when it meets all the requirements it gets in touch with the certification lab to prove that it meets all the requirements provided by the SCMS provider. And the certification lab is trusted by the SCMS provider to carry out its steps for the security requirements on the device.

So, a little bit in more detail, the deployment manager is responsible for selecting and contracting with the SCMS provider. The deployment manager also determines the device suppliers it is going to use in the deployment. And finally, it is also responsible for determining the set of Root CA certificates that the device should be trusting in the system in a deployment. The SCMS provider, on the other hand, states the certificate and security requirements for applications in the deployment. This may include a requirement for third-party type certification by a certification labs. And when necessary SCMS provider provides a list of certification lab that is approved or accredited.

An SCMS provider works with deployment manager to specify certificate and security policies for also deployment specific applications. It is in charge of running the enrollment CA, Registration Authority, which are the points of contract for devices as described in the last learning objective. It is also running the authorization CA. And in future all these different CAs and the RA it should be possible to get these from different providers. The device supplier provides devices that satisfy the requirements specified by the SCMS provider. So, it provides devices with 1609.2 and SCMS client software. It is responsible for meeting all the security and interoperability requirements. And to demonstrate that it meets all those requirements it may need to interact with a certification lab. And finally, the certification lab works with the device supplier to determine which devices meet the requirements. Certification labs are by definition trusted by SCMS provider. But they are not necessarily in direct contact with the deployment manager, but they may interact with them from time-to-time. There are

currently some certification services provided in the US by the OmniAir Consortium. For more details please follow the next learning objective.

So, having covered all the four different types of parties interacting in the SCMS in our expert opinion, we would like to note that we highly recommend using a third-party SCMS provider for SCMS certificates. And the reason for that is that running an SCMS component or several components is not easy, especially for components like Root CAs, where you need to you need to take care of so many different issues including secure key storage. You would be surprised to know that some SCMS providers use a vault in the mountains to store Root CA keys. The graphic on the right side of the slide demonstrates that. Other than secure key storage, there are issues like waking the CA up whenever needed. Then you also need to worry about how to enforce the different policies that are set by the SCMS provider. It also needs to follow the audit requirements. And finally, it has certain liability issues associated with it.

In conclusion, SCMS should be run internally only if the manager—the deployment manager—has significant experience running public infrastructure, and knows and realizes all the risks associated with running such complicated systems. So with that let's look at the interactions between devices and the SCMS. In this system, all the devices that use certificates they need to come fully equipped with certificate management software, the software that the devices can use to request and receive certificates from the SCMS. This was discussed in the key point one in this learning objective. Deployment sites in many cases they may also want to have centrally generated messages, for example MAP messages or traveler information messages. The graphic in this—in the below slide—in the bottom part of the slide shows these two centrally generated messages. So the map is generated and signed at a Map Generation Center and then forwarded from traffic—from Map Generation Center to Traffic Management Center and onto a connected vehicle roadside unit.

Similarly, a traveler information message is generated and signed at the Traffic Management Center and then forwarded to a connected vehicle roadside unit for dissemination to the connected vehicles. A few important points about centrally generated messages. These centrally generated messages should be signed at the point of generation, rather than at the point of distribution. And the reason is pretty simple. If these messages were signed at the point of distribution so, for example, if there was a compromise roadside unit, and it was allowed to sign these messages, it could easily flood the message—flood the system with fake messages. These centrally generated messages have unique requirements and therefore the deployment manager needs to have a special arrangement with the SCMS provider to set the right security requirements.

For example, is it enough to generate the message on a box at the Traffic Management Center with a smart card? Or does it need an appliance with heightened security? And then what are the network security requirements to ensure correct operation on the box at the Traffic Management Center? We would like to note that there have been existing deployments that have successfully used certificates for centrally generated messages. So this problem can be addressed. We need to pay special attention to centrally generated messages because of the risks involved. So finally, if centrally generated messages will be part of a deployment then security requirements for these messages should be addressed early in the process.

With that, we have come to the end of this learning objective, and let's look at a question for this learning objective. So the question is what is the minimum number of Root CAs that must be supported in a deployment? And the four choices are zero, one, two, and all existing Root CAs. If you need more time to review the answers, please feel free to pause the presentation.

So the correct answer is one. Let's look at the incorrect answers. So the first incorrect answer is zero. If there are no roots CAs that are trusted in the system then the system cannot operate. Two devices cannot trust each other. The third answer is also incorrect because a system can operate just fine with just one Root CA. So we don't necessarily need two Root CA in the system to be trusted. And the fourth option is also incorrect because to start with you don't need to adjust all existing Root CAs. When the electoral system will be stood up devices can automatically trust all Root CAs, but this is not necessarily a requirement for a deployment.

So that concludes the first part of this course. In summary, we learned in the first module in the first learning objective how to define communication security requirements in the connected vehicle environment. Then in the second learning objective, we went over the different parts of the Security Credential Management System, and the different cryptographic building blocks that it uses to provide trust in the system. And finally, in the last learning objective of this part of this course, we learned how the devices in the system get in touch with the SCMS to start interacting in the system.

So with that I would like to quickly go over the second part of this course which will have the two remaining learning objectives. The first learning objective for the next part of the course is to identify the vehicle-to-everything or V2X for short certification process for a device to enroll in the Security Credential Management System. And finally, in the last learning objective, we will learn how to make a deployment plan that uses SMS services. So with that this part of the course is concluded. And thanks a lot for completing this module. Your feedback is very welcome. Please use the feedback link

below to provide us with your thoughts and comments about the value of the training. Thanks a lot.