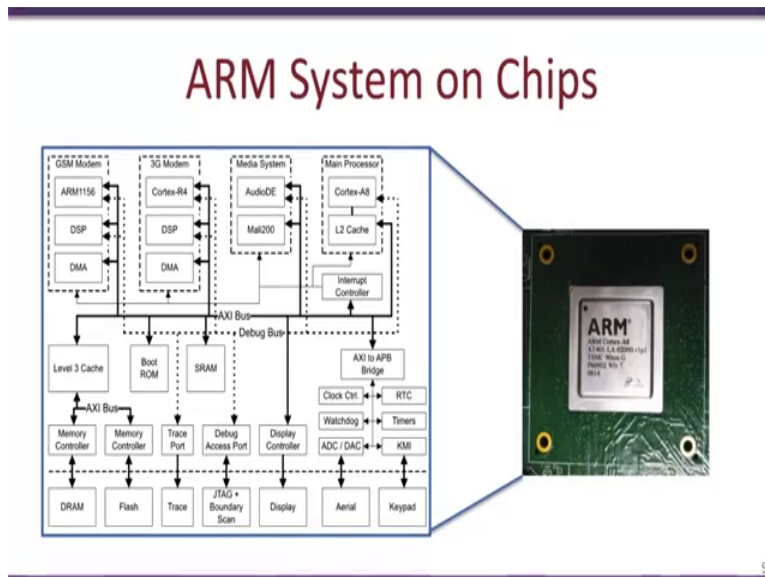


**Information Security - 5 - Secure Systems Engineering**  
**Professor Chester Rebeiro**  
**Indian Institute of Technology, Madras**  
**ARM Trustzone**

Hello and welcome to this lecture in the course for Secure Systems Engineering, so in these sequence of lectures we have been looking at Trusted Execution Environments in this lecture we will be looking at the ARM trustzone so a lot of this information that we covering in this lecture can be obtained from this link which is about the trustzone architecture.

(Refer Slide Time: 00:44)

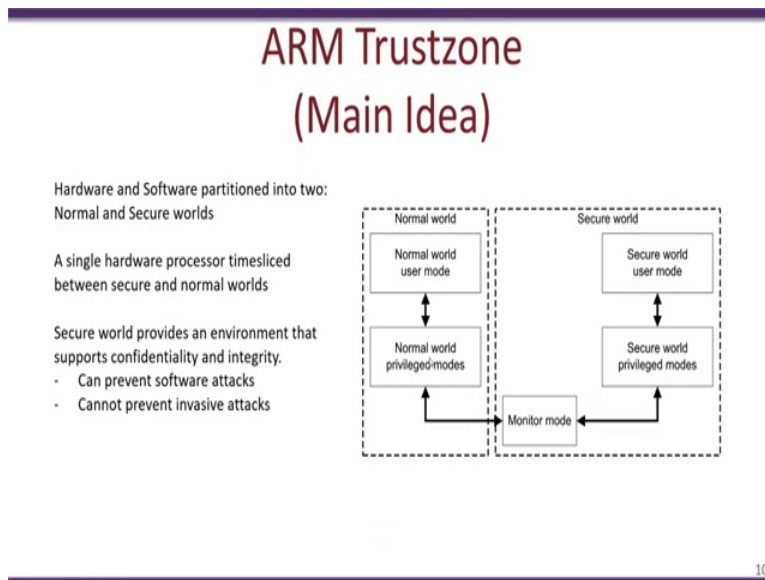


Before we go into the ARM trustzone we will take a look at what the system on chip means. So a typical system on chip would look like something like this it could have various components corresponding to the various (function) to different functionality.

So you would have example the main processor you have a lot of different modems GSM 3G modem so this is shown for a typical say a mobile application and you would also see various other components such as ADC, DAC, timers and so on, so all of these things would be present in a single chip. Now the advantage of having such a system on chip architecture is that a size of your complete design is reduced considerably. So for example if you take the case of a mobile phone and you actually open up your mobile phone you would see that there are very few IC's present on it and the reason being is that each of this IC's have a lot of different functionality.

So in prior years previous prior to the system on chip type of architecture you would have a different IC present for each of this modules and therefore the size of your entire design would be quite large right. Now with the advent of the system on chip and lot of all of this components put into a single chip the size has reduced considerably and this size actually cost a large number of different opportunities for example the size of mobile phones etc have shrunk thanks to this particular technology. So now we go into a big more detail about the ARM trustzone.

(Refer Slide Time: 02:39)



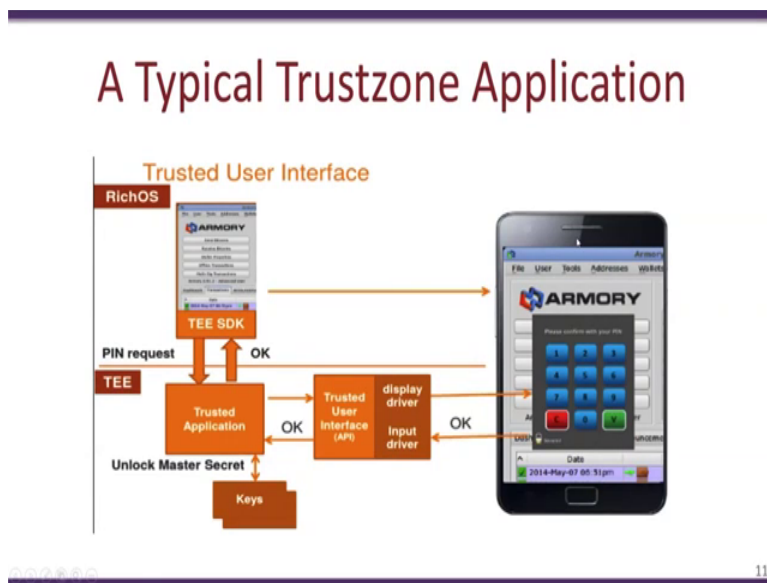
So the ARM trustzone essentially creates two partitions. It creates a normal world and a secure world so the figure looks something like this so in the normal world is where you would have your typical operating system like your android operating system on IOS and you would be running your typical pass like your whatsapp or anything else so all of them run in this normal world. Now you would have a secure world as well which would run your sensitive task so all your sensitive operations such as for example encryption or entering sensitive data like passwords or OTP numbers would be handle by the secure world.

So for example you could consider this right so you would have normal world applications and whenever for example you require to do some sensitive operation the processor shifts to the secure world you enter your password which gets authenticated or you enter your OTP or do an encryption which further gets verified and then once it is verified you switch back to the normal world. So ARM provides this particular environment and provides and manages this switching

between normal world and secure world I think hard work is built in such a way that the normal world applications will not be able to access or is totally isolated from the secure world applications and therefore all the sensitive information and secure computations which is done in the secure world is completely isolated and completely independent of the normal world operations.

So thus any software vulnerability or any malicious code any malware present in the normal world cannot steal information ok not affect the secure world operations.

(Refer Slide Time: 04:44)



So just to take an example of how it would typically look like so let us say we have an ARMORY application as shown over here so the ARMORY application runs in the normal world so we call the operating system present in the normal world as the Rich OS. So your typical android operating system or the IOS operating system is the Rich OS so we call this the rich OS because it is filled with various features and most of your applications would be running in this rich OS and making use of the various features that are supported by that particular operating system.

So let us say we have this ARMORY application running from our Rich OS and there is one button over here which says that the user has to enter a PIN so when this button is pressed there is a switch from the Rich OS that is in the normal world to the secure world and in such a case and there would be a execution in the trusted environment. So in this particular mode or the

secure world it would pop up a window like this and request the user to enter a PIN so all of this transactions in the secure world is completely isolated or completely done securely and not accessible from any of the applications present in the normal world.

Even the android or IOS switch OS running from your normal world would not be able to have access to the PIN which is entered by the user. So once the PIN is entered through the display driver and the input driver there would be a trusted user interface which actually reads this information and pass on to the trusted application which then authenticates the PIN using keys which are stored in the trusted environment if the authentication is right then the and ok message is sent back to the normal world application that is the ARMORY application and the user of this application would then continue to use this application.

So what you see over here is that because of the trustzone which is supported by the ARM all the activities all the keys all the authentication which is done in this trusted environment I mean the secure world is completely isolated from the user world applications. So the keys and so on are secure in spite of any malicious code that is present in the normal world system so even if your android or IOS is compromised still the key is stored in the trusted environment is still safe and secure.

(Refer Slide Time: 07:43)

---

## Switching Worlds

- Execution in time sliced manner (Secure <-> Normal)
- New mode (monitor mode) that is invoked during switching modes
- Mode switching
  - triggered by *secure monitoring call* (SMC) instruction
  - certain hardware exceptions (interrupts, aborts)
- **Monitor Mode:** saves state of the current world and restores the state of the world being switched to. Restoration by *return-from-exception*.
- NS Bit: in configuration register indicates secure / normal operating mode.  
NS = 1 -> indicates non-secure (normal) mode

---

12

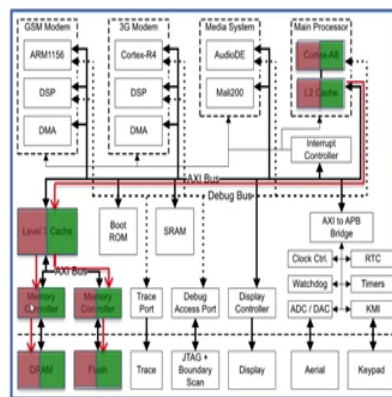
So we look a little bit about how ARM actually manages this to have two environments secured and the normal world environment within the same processor. So essentially this two environments work in a time sliced manner so there is switch from the secure world to the normal world and vice versa further there is a new mode of operation that has been introduced known as monitor mode which essentially gets invoked whenever there is a switch from one mode to another on return so there are two ways in which this mode switching is triggered by first way is by software by an instruction known as the secure monitoring call or this SMC instruction second you could also have hardware interrupts or exceptions which occur and this could also be to switching from of modes from normal to secure.

Now the monitor mode is a crucial role during this mode switching first what it does is that it saves the state of the current world that is either normal world or the secure world and restores the state of the world that is switched to so the restoration is done when there is a return from an exception. So for example whenever there is let us say that an interrupt occurs the first thing that gets executed is code present in the monitor mode so the monitor mode within identify the interrupt that has occurred it would save the context of the current world that is if when application is running in the normal world the registers corresponding to that particular application is saved in the monitor mode and then there is a context switch to the secure world and the interrupt routine corresponding to that particular interrupt is then executed.

After that interrupt service routine is executed and the interrupt gets handled there is a return from exception instruction that gets executed now this would result in the monitor mode getting executed again and then the monitor would restore the state of the application that is executing. Now in order to keep track of the mode of operation the ARM has a particular bit known as the NS bit so the NS bit present in the configuration register indicates whether it is a normal world or the operating world that is currently being executed. So for instance if the NS bit is equal to 1 it indicates that the processor is in the normal world, if the NS bit is set to 0 that would indicate a secure world operation.

(Refer Slide Time: 10:33)

## NS Bit extends beyond the chip



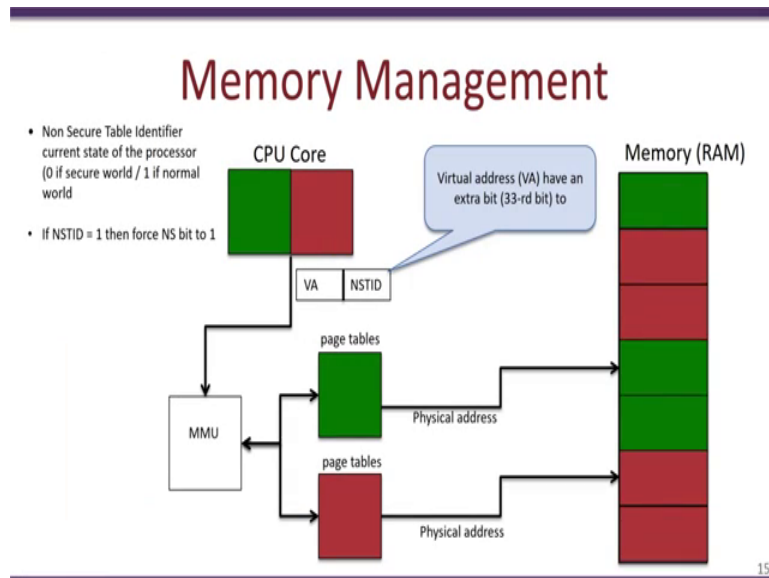
14

Now the NS bit is present in a configuration register or present in the main processor and based on the value of the NS bit the processor over here the cortex A8 works in either the normal mode or the secure mode similarly the cache present in the processor is also able to distinguish between memory accesses which are for the normal world and the secure world but there is a lot more things that happen when there is this mode switch one important thing for us is that this NS bit also extends outside this processor component so for example over here we show that the red line indicates that the mode of operation is also transferred to other components present in the system on chip.

So thus other components would also be able to distinguish between a secure world operation and a normal world operation so for example and important thing for us or we will see later is

that memory controller present in the system on chip would be able to distinguish between memory access which is made to a normal world operation and a memory access made to a secure world operation.

(Refer Slide Time: 11:48)



Now so we look at a little more detail about the memory management present with trustzone as we have seen that the CPU core that is a cortex A8 will be switching from the secure world and the normal world and the NS bit present in the CPU configuration register would identify which world is executed. Now every time there is load of store instruction from one of this codes either the secure world or the normal world mode of (ope). So whenever there is an instruction from one of these worlds either the secure world or normal world. What is done is that there is a virtual address put out on the bus so this virtual address goes into the memory management unit so a typical 32 bit ARM core would put out a 32 bit virtual address on this particular bus. Now with trustzone enable processors an additional bit known as the NSTID bit is 35 bit put the also put out on the bus so this particular bit would identify the current state of the processor.

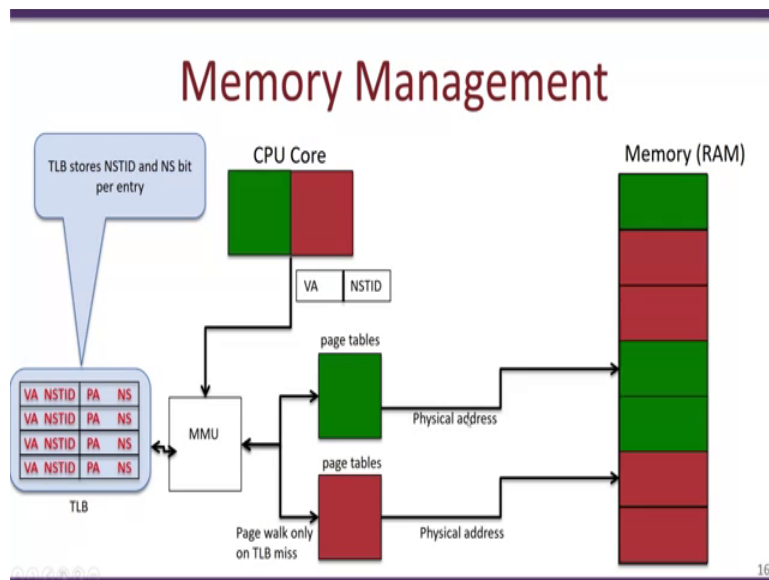
It would have a value of zero to indicate that the load of store operation or the instruction that is requested is from the secure world if it is 1 that bit would indicate that the load of store instruction fetch would be from a normal world operation. If the NSTID bit set 1 then the NS bit is forced to 1 now this because of this additional NSTID bit which is sent to the MMU the MMU would be able to identify or distinguish between secure world load or store request and a normal

world load or store request. So based on this NSTID bit the MMU would be able to select page tables or use page tables which are meant for the secure world or the normal world ok.

Now the page tables convert the virtual address to corresponding physical address and each of this page tables would thus be able to point to pages which correspond to secure world pages or the normal world pages so for example if there is a say load or store request to a particular virtual address the NSTID bit is set to 1 which would indicate that it is a normal world memory request, what the MMU would do is that it would use the normal world page tables. So the virtual address would then get translated to the corresponding physical address and therefore the mapping is done in such a way that it is only the normal world pages which can be accessed.

Now if this particular virtual address falls in one of this normal world pages in the RAM then the load or store will be successful. On the other hand if a virtual address is put out on a bus which actually corresponds to a secure world page then there would be a trap and the operation would not be permitted.

(Refer Slide Time: 15:07)



Now the MMU as we know also comprises of TLB which is the translation look aside buffer now in an ARM core which has trustzone enabled in it the TLB is modified slightly. Now as we know the a typical memory management unit also has a TLB present in it. The TLB it stand for the translation look aside buffer has a mapping between the virtual address and the corresponding physical address so this mapping will ensure that once a virtual address is mapped

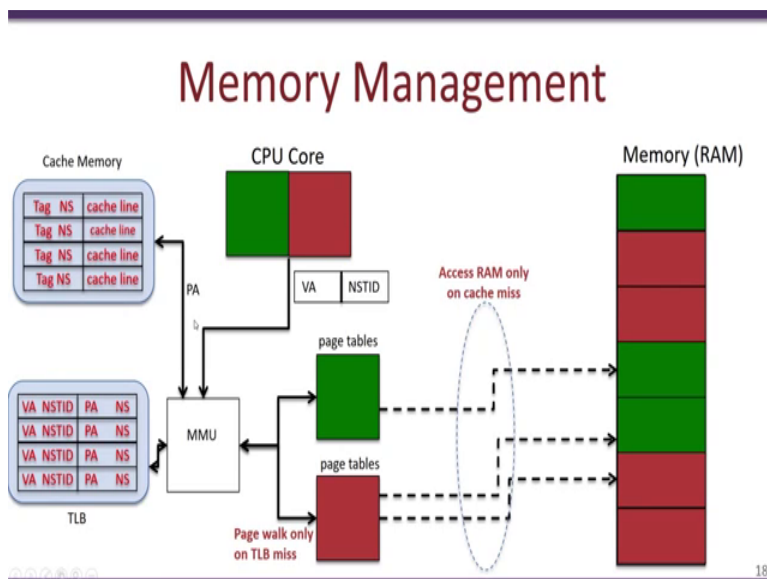


to its corresponding physical address using the page tables that are present that mapping from virtual address to physical address is stored in the TLB. So the TLB would be fully associative cache and it is expected that if the locality of the execution and memory accesses you are quite lucky to find the mapping of virtual to physical address present in the TLB and a lot of overheads will be reduced.

Now in an ARM processor which is enabled with the trustzone the TLB is modified a bit now each entry of the TLB not only comprises of the virtual address and the corresponding physical address but also has details about the NSTID bit corresponding to the virtual address and the NSBIT corresponding to the physical address so in other words it will identify whether the virtual address was need with respect to a normal world operation or the secure world operation. Further it also has information to say whether the physical address corresponds to a page which is secure or in the normal world.

So based on this TLB which is stored trustzone enable ARM systems the MMU would be able to use the TLB to say permit a secure world page table to access secure pages as well as normal world pages, why having this particular TLB? The MMU would be able to use the TLB to allow secure world virtual address for MMU would be able to use the TLB to permit a secure world page tables to access normal world page on the other hand it can also prevent a normal world page table from accessing a secure world page.

(Refer Slide Time: 17:43)



Now in addition to the MMU the cache memory present in trustzone enabled ARM processors is also modified. Both secure world as well as normal world memory request are stored together in the same cache however there is an additional bit known as the NS bit which is stored in the tag however the tag is modified so that each tag also comprises of the NS bit it thus based on the tag and the NS bit present a memory request can be identified whether the corresponding cache line corresponds to a secure world cache line or a normal world cache line.

(Refer Slide Time: 18:24)

---

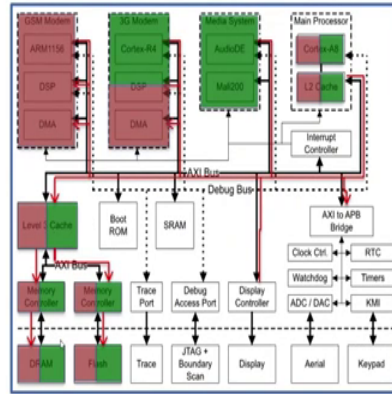
## Memory Management Units

- Two virtual MMUs (one for each mode)
  - Two page-tables active simultaneously
- A single TLB present
  - A tag in each TLB entry determines the mode (Normal and Secure TLB entries may co-exist; this allows for quicker switching of modes)
  - alternatively the monitor may flush the TLB whenever switching mode
- A single cache is present
  - Tags (again) in each line used to store state
  - Any non-locked down cache line can be evicted to make space for new data
  - A secure line load can evict a non-secure line load (and vice-versa)

Now in order to make this things to work the memory management unit in trustzone enabled ARM processors is designed in a special way.

(Refer Slide Time: 18:40)

## Secure and Normal Devices



20

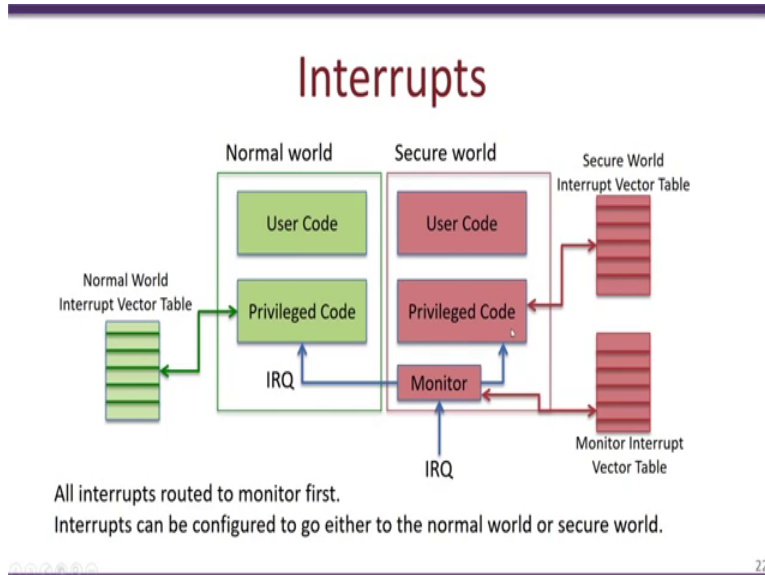
So for example there are two virtual memory management units that are present in addition to the memory management unit which we have seen the NS bit which determines whether the processor is running in secure world or normal world and they actually send to all other components present in the system.

Thus one would be able to configure a particular component in the system to work only from a particular mode for example over here what we see is that the GSM modem, 3G modem and the media system is configured differently so the GSM modem is configured so that it works only when the main processor is running in the secure world. On the other hand the 3G modem is configured to work both from the secure world as well as the normal world putting this in other words when the main processor is running in the normal world it will not be able to access the GSM modem thus applications running in the normal world would not be able to even see that the GSM modem is present in the SOC and no communication to the GSM modem is possible.

On the other hand when the main processor is in the secure world or configured for the secure world then the GSM modem the 3G modem and all other devices would become accessible so this technique is essentially useful for example where you are able to configure say the keypad to only work in the secure world. So this would permit that whenever we enter passwords or prints or anything else so this passwords and prints are only accessible through a keypad which is

enabled during the secure world of operation and thus it is also secure from any kind of snipping attacks.

(Refer Slide Time: 20:27)



So interrupts are a critical aspect with respect to trustzone so as we have seen that is a monitor mode present over here so the monitor is part of the secure world and what we see is that whenever interrupt comes so it is first channeled to the monitor mode. The monitor mode which say the context of the current executing environment and then decide whether that interrupt can be should be serviced by a normal world application or a secure world application. So based on how this various interrupts are configured this interrupt request would be either channeled to the normal world interrupt service routine or the secure world interrupt service routine. So each of this worlds would have its own interrupt vector routine which would then look up the corresponding interrupt request determine where the interrupt service routine is present and then execute that corresponding interrupt service routine.

So for example we have privileged code over here could be at android OS so let us say for example that a network interrupt occurs and a network interrupts are configured to be handle by the normal world. So therefore the monitor would channel the network interrupt to your privileged code which essentially could be at android operating system your android OS would then look up the normal world interrupt vector table identify the interrupt service routine corresponding to the network interrupt and then execute that corresponding service routine.

On the other hand if the interrupt happened to be a secure world interrupt the monitor would then channel that secure world interrupt which would then look at a secure world interrupt vector table and identify the corresponding location for that interrupt service routine.

(Refer Slide Time: 22:15)

---

## Software Architecture

- The minimal secure world can just have implementations of synchronous code libraries
- Typically has an entire operating system
  - Qualcomm's QSEE; Trustonics Kinibi; Samsung Knox; Genode
  - The secure OS could be tightly couples to the rich OS so that a priority of a task in the rich OS gets mapped accordingly in the secure OS
  - Advantage of having a full OS is that we will have complete MMU support
- Intermediate Options

We now look at how a typical secure world software looks like, a typical secure world software would have implementations of very minimalistic implementations of synchronous code libraries. So typically we would have operating system present in the secure world so this are specialized operating systems which have very minimal functionality.

So operating systems such as Qualcomm's QSEE, Trustonics Kinibi, Samsung Knox, Genode etc are secure world operating systems. So this operating systems are tightly coupled to the corresponding rich operating systems so that a priority of a past in the Rich OS can get napped to a corresponding priority in the secure OS. So examples of having a full OS is that it will have complete.

(Refer Slide Time: 23:09)

---

## Secure Boot

### Why?

Attackers may replace the flash software with a malicious version, compromising the entire system.

### How?

Secure chain of trust.  
Starting from a root device (root of trust) that cannot be easily tampered

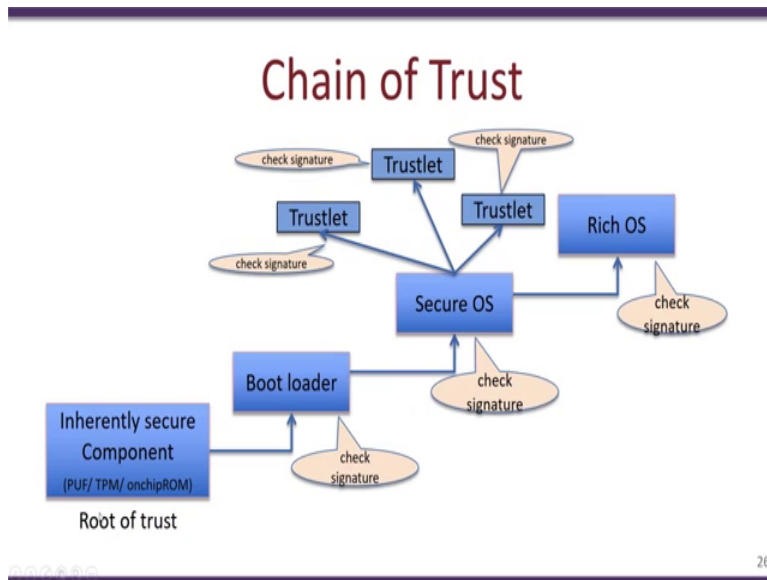
---

24

One thing that is critical with respect to a trustzone is something known as secure boot. The reason why we require some secure boot is the following, so let us say that we are having an application that uses the secure world.

So as we know that the application would be stored in the flash and what could possibly happen is that an attacker could modify the flash contents and therefore could modify the secure components of that application the function maliciously. So for example an attacker would insert Trojan's or any other malware in the secure component of the application thus we need to ensure that no secure world software gets tampered with while storing in the flash and one way to achieve this is by using secure boot. The way secure boot works is by something known as a chain of trust so starting from the root device there is a trust created in the system based on this particular mechanism it becomes very difficult to tamper with.

(Refer Slide Time: 24:24)



A typical chain of trust looks something like this. We start with a particular component known as the root of the trust this could be a physically (( ))(24:29) function on chip ROM or a trusted platform module or a TPM so this is the basic or the root of your trust so it is assume that this is your trusted module so this trusted module would then first verify that the boot loader has not being manipulated so to do this by checking that the digital signature of your boot loader has not being tampered with so this root of trust at the time of boot first starts to execute and verifies that the boot loader has not been tampered with.

So it would do this verifying that the signature of that boot loader is correct so this could be for example (( ))(25:18) or even more complicated digital signatures to unsure that the boot loader has not been manipulated or not being tampered with. So if it determines that the boot loader has is indeed not tampered then it would pass on execution to the boot loader and the boot loader with then execute. After the boot loader is nearly completing its execution and would want to boot the secure OS it could first determine that the signature of the secure OS is correct this can be verified but checking the footprint or by computing the signature of the secure OS present in the flash and verifying this particular signature with a stored signature.

If it is found that signature of the secure operating system does not match the signature which is stored then the secure OS is not (( ))(26:14) on the other hand if we find that there is a match then the boot loader would has actually verified that no tampering has occurred on the secure OS and

would it could permit the secure operating system to boot. Now the secure operating system would correspond various trustlet and before (( ))(26:35) any of this trustlet is would as we seen before identify that each of this trustlet have not being tampered with. So in this way various trustlet are created only after their signatures are authenticated.


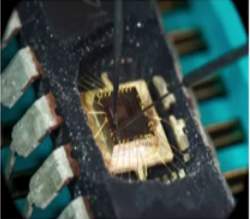
In a very similar way the secure OS initiates the boot of the rich operating system only after validating that the signature of the rich operating system is correct so in this way what we see is due to the presence of a root of trust we build a chain of trust. First we obtain trust in the boot loader then we obtain trust in the secure OS and from the, the rich OS trustlet and so on.. Now if any of this modules are tampered with in the flash this can be detected by the secure chain of trust the only assumption that we make is that this root of trust cannot be tampered with that is the only assumption that we make.

(Refer Slide Time: 27:39)

---

## Points to Ponder

Describe how ARM trustzone can handle invasive attacks? What can it handle? What are the limitations?	Why is the monitor part of the secure world?
---	--



---

Another thing to think about is if you go back to the block diagram we see that the monitor is part of the secure world so could you think about what is the rational for having the monitor as part of the secure world. In the next lecture will look at the Intel SGX trusted execution environment, thank you.