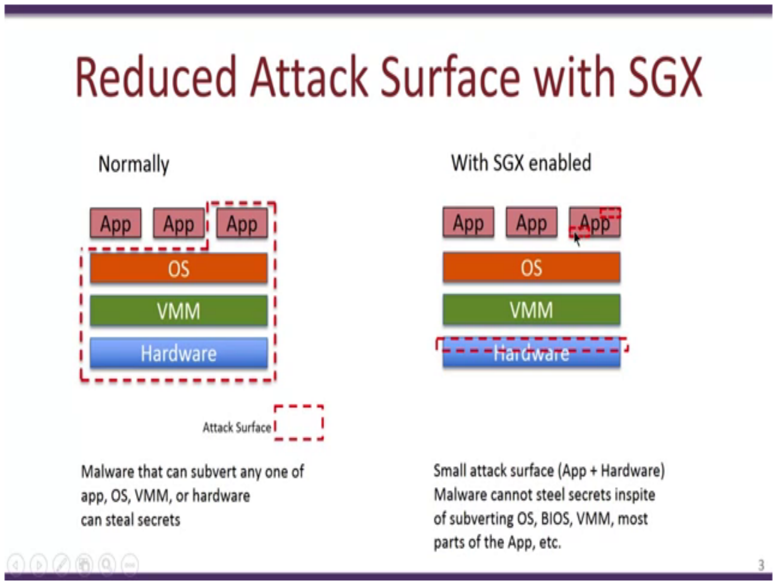
Information Security - 5 - Secure Systems Engineering
Professor Chester Rebeiro
Indian Institute of Technology, Madras
Intel's SGX Part 1

Hello and welcome to today's lecture in the course for secure systems engineering so in today's video lecture will be looking at Intel's SGX Software Guard Extensions. So this is part of series of videos on trusted execution environments we looked at the ARM trustzone in the previous video and the ARM trustzone also creates a trusted execution environment but as will see in today's video about SGX the entire way of doing it would be much different therefore from some perspective the guarantee is obtain and the security obtained by SGX would be much more preferred for server type of machines while ARM's trustzone is more suited for the embedded platforms so on.

(Refer Slide Time: 01:13)



Reduced Attack Surface with SGX

Malware that can subvert any one of app, OS, VMM, or hardware can steal secrets

Small attack surface (App + Hardware) Malware cannot steel secrets inspite of subverting OS, BIOS, VMM, most parts of the App, etc.

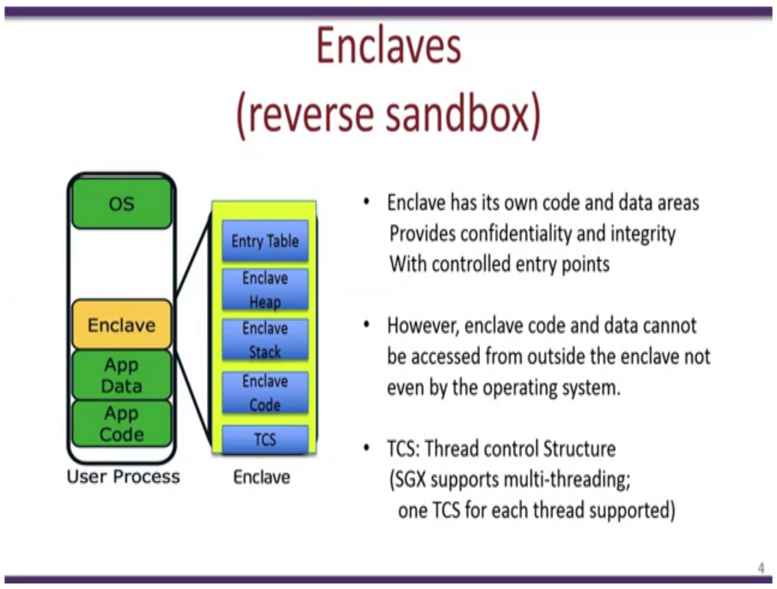
When we actually normally look at a machine or a system we see that there are a multiple layers of hardware there are VM's, operating systems and above that the application. Now the trust surface or the attack surface in this particular scenario is this entire scheme. So what we mean by this is that if let us say the application has something secret let us say a secret key then this particular boxed area are is the region which you actually assumed to be trusted in order to keep that key secure, for instance if there is a malware in the application then that particular malware could steal that secret key.

Similarly if a malware now attacks the operating system or the VM then the key which is present in the application can be compromised. In a similar way a malware or a Trojan present in the hardware could steal the secret key in the application. So what we see over here is that in order to assume or keep something secret in a normal system we would require a huge amount of assumptions that all the underlined hardware the system software compromising of the virtual machines, managers and the operating system are all trusted. Now what SGX actually helps us do is that it reduces the potential attack surface so this is how SGX enable the computer system would look like and over here let us say we still have an application and this application has a secret key.

Now with SGX this secret key would only require that portions in the application which is boxed in this red dotted line and portions in the hardware is actually trusted. So you could still have an untrusted operating system untrusted virtual machine managers and so on but what is required keep the key secret is just that the hardware a portion of the hardware is trusted along with small areas of the application. So what we achieve by this is that we are drastically reducing the attack surface. Now if you compare the normal mode of operation versus the SGX enable mode of operation we see that an attacker could have attacked either the application OS, virtual machine or the hardware in order to gain access to that secret key.

While in the SGX enabled hardware the attacker would have to target small regions in the hardware or small portions of code in the application in order to achieve in order to gain access to the secret key so let us look into more details about how SGX actually works.

(Refer Slide Time: 04:11)



Enclaves (reverse sandbox)

OS
Enclave
App Data
App Code
User Process
Entry Table
Enclave Heap
Enclave Stack
Enclave Code
TCS
Enclave

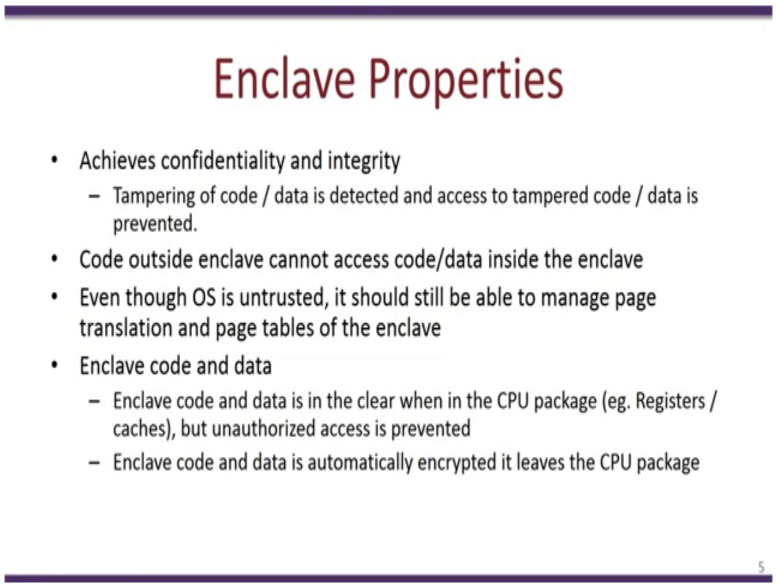
- Enclave has its own code and data areas
  Provides confidentiality and integrity
  With controlled entry points
- However, enclave code and data cannot be accessed from outside the enclave not even by the operating system.
- TCS: Thread control Structure
  (SGX supports multi-threading;
  one TCS for each thread supported)

4

So let us say that this is our process space so as we have seen before the process space has the code that is the application code application data comprising of stacks heaps and so on and higher in the typical address space of a process is where the operating system resides. Now what SGX permits us to do is to have enclaves in this address space so what we see over here is that within this entire address space of the process there is one region which is called the enclave which provides the necessary sandbox to achieve the trusted execution environment. So within this particular enclave you could have some regular code stack and heap so we call this as the enclave code, enclave stack and the enclave heap and you would have some management data such as the entry table and TCS which is a Threat Control Structure so all of this can be present in an enclave.

Now what makes this enclave unique is that any code, any function or anything which is executing outside this enclave for example in this green region over here will not be able to access any code or data present within the enclave. So for example a code present over here cannot directly call a function present in the enclave code. Similarly code present outside the enclave will not be able to directly invoke data or access data in the stack or in the heap present in the enclave. However the vice versa is true now if you are running code within the enclave this particular code will be able to access the stack and heap present in the enclave as well as all the other application data and code present outside the enclave as well.

(Refer Slide Time: 06:08)



Enclave Properties

- Achieves confidentiality and integrity
  - Tampering of code / data is detected and access to tampered code / data is prevented.
- Code outside enclave cannot access code/data inside the enclave
- Even though OS is untrusted, it should still be able to manage page translation and page tables of the enclave
- Enclave code and data
  - Enclave code and data is in the clear when in the CPU package (eg. Registers / caches), but unauthorized access is prevented
  - Enclave code and data is automatically encrypted it leaves the CPU package

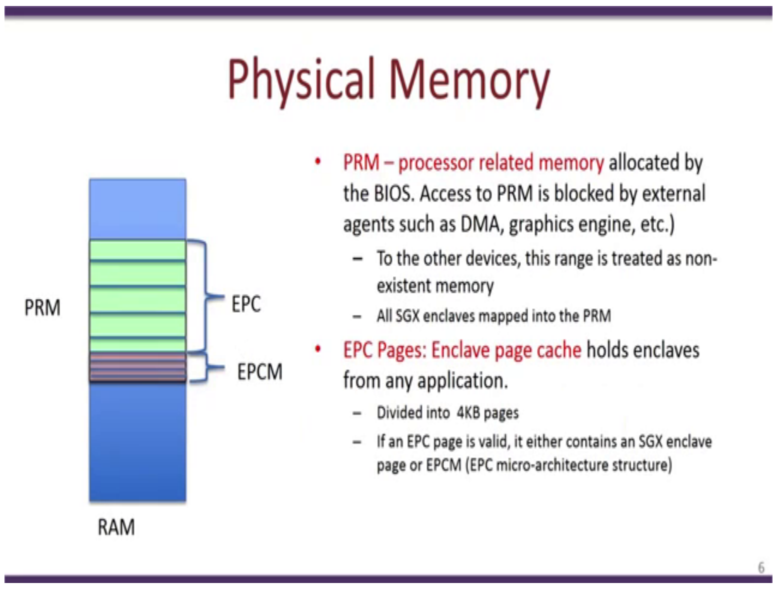
5

So what the enclave actually permits us to do is that it would it is able to create a closed sandbox through which you could get confidentiality and integrity so what we mean by confidentiality is that the code and data present inside the enclave is kept confidential with respect to anything or any code or anything else outside the enclave. So typically this is achieved by multiple ways for example there is encryption which is done and will see more about it later in the video and also there is special memory management units present in the hardware which creates an environment so that confidentiality of the code and data in the enclave is achieved.

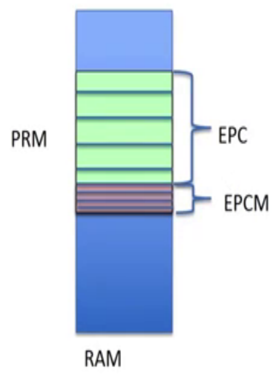
In a similar way we also achieve integrity of the code and data that is present within the enclave so what we mean by this is that any code and data cannot be tampered or modified by an external party so this is typically done again using special purpose hardware, special purpose memory management units present in the hardware as well as a lot of cryptography algorithms so ensure that the data is not tampered with. Now one thing what the SGX in the Intel processors also achieve is that it ensures that when data pose outside the processor that is if data or code from the enclave is going in or out of the processor it would has provisions to ensure that no attacker could actually monitor the various buses or snoop at the D-RAM present on the system and would be able to actually identify what the code and data actually is or this is achieved by having memory encryption.

So, essentially any data which is to be executed from the enclave is going to be stored in the RAM in an encrypted state. Now when this data is moved from the RAM to the processor it gets decrypted within the processor. So thus if there is a snooping done on the data bus or there is actually snooping within the D-RAM then what would happen is that the attacker would only see encrypted code and the sensitive data and information is still kept secret.

(Refer Slide Time: 08:30)



Physical Memory



- PRM – processor related memory allocated by the BIOS. Access to PRM is blocked by external agents such as DMA, graphics engine, etc.)
  - To the other devices, this range is treated as non-existent memory
  - All SGX enclaves mapped into the PRM
- EPC Pages: Enclave page cache holds enclaves from any application.
  - Divided into 4KB pages
  - If an EPC page is valid, it either contains an SGX enclave page or EPCM (EPC micro-architecture structure)

6

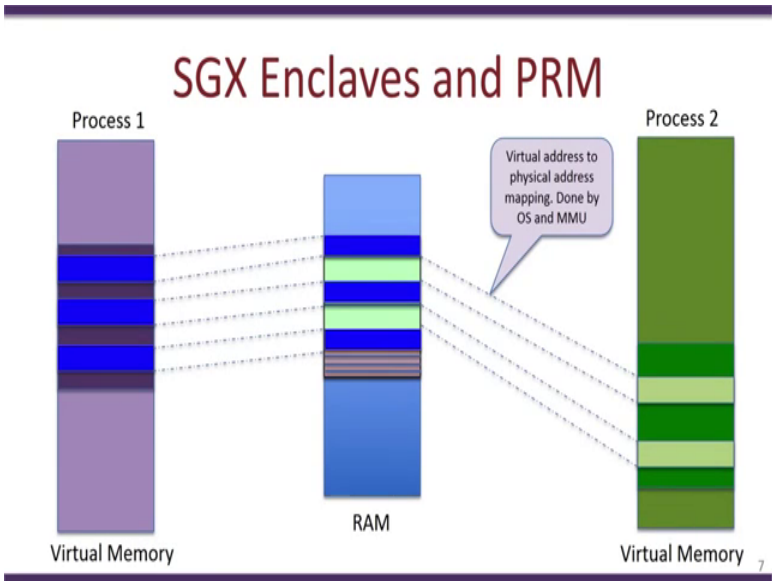
So let us see how these things actually work internally. So SGX is quite a complicated mechanism to achieve this trusted execution environment and we will just see a very brief overview in this lecture about how this things would work.

So to start off with what happens in an SGX enabled Intel processor is that some portion of the RAM is marked as PRM so this is known as a Processor Related Memory so this area let us say some from some address A to some address B is marked by the BIOS as a processor related memory so what this means is that the operating system or any applications running over the processor would not directly be able to access the memory in the PRM also the specialty of this PRM region of memory is that no external device like no external master device such as network guards or graphic engines and so on would be able to read or write the data to this PRM essentially going a bit more technical the DMA accesses to this particular region of memory that is PRM region of memory is disabled.

So what this means from an operating system is that this region the PRM region is identified by the OS as non-existent memory so it means that there is a hole in the entire memory RAM's of the processor or rather the operating system sees this PRM as a hole in the memory region and therefore would not allocate any memory or any other data or try to store any data in this particular region. From this what we actually achieve is that we have this region of memory which is completely in the control of the hardware so the hardware could use this region of memory to create enclaves, managed the various enclaves, manage the memory who accesses this enclaves the read write execute permissions for this enclaves and so on.

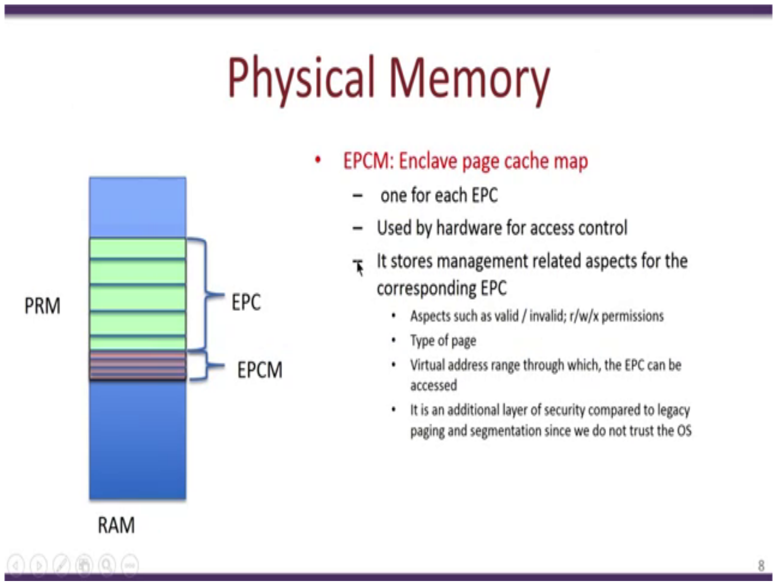
So internally what happens with the PRM is that it is divided into several EPC's or enclave page caches. So each of this page caches which is shown over here is of 4 kilo bytes so this page caches are used to actually store the enclaves of the various applications present in the program and also there is some management related information which is stored in a special region known as the EPCM or which expands to EPC micro architecture.

(Refer Slide Time: 11:15)



So the way it works is that we have multiple processes present in the system each process has its own virtual address space and within this virtual address space each process could have its own enclave. So per process you could have one or more enclaves or you could also have a process without any enclaves as well. So therefore you would now have a mapping for this enclave region within the virtual address space to this PRM region in the RAM. So we have the memory management unit which converts the virtual memory to the physical memory address and the operating system would ensure that addresses form this enclave gets mapped to physical pages within the PRM present in the RAM. So therefore you could actually have multiple processes like this each of them having their own enclaves but all of this processes would mapped to the same region within the Ram that is the PRM region.

(Refer Slide Time: 12:19)



Physical Memory

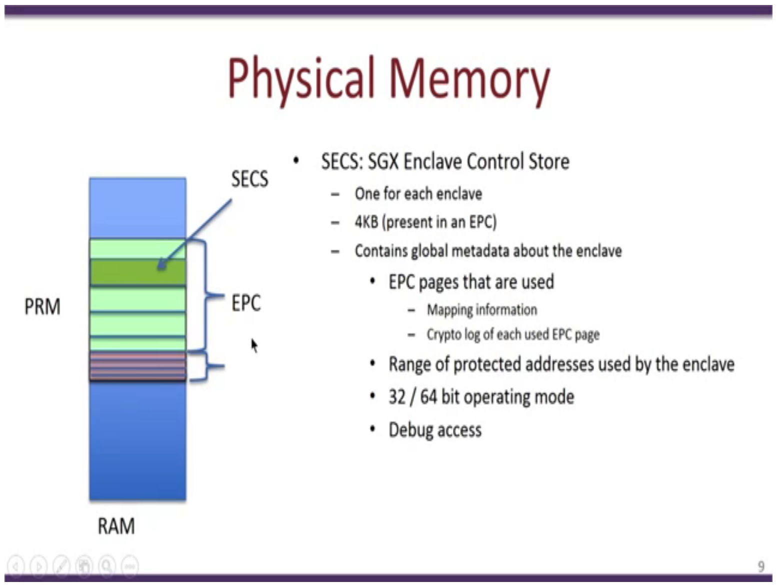
- EPCM: Enclave page cache map
  - one for each EPC
  - Used by hardware for access control
  - It stores management related aspects for the corresponding EPC
    - Aspects such as valid / invalid; r/w/x permissions
    - Type of page
    - Virtual address range through which, the EPC can be accessed
    - It is an additional layer of security compared to legacy paging and segmentation since we do not trust the OS

So let us look at the EPCM which is the acronym for enclave page cache map so this EPCM is further divided into various sub-regions and we have one entry for each EPC so for example if there say a 1024 EPC regions each of 4 kilo bytes then there would be 1024 regions in the EPCM, one region is associated with a corresponding EPC so essentially this EPCM is used by the hardware for access control it stores various information related to the corresponding EPC so for example this particular EPC would have a corresponding EPCM entry which stores aspects such as the validity or of that particular EPC the read write execute permissions for this EPC for example if there is code present over here you would have that particular code as executable and not writeable.

It also stores the type of page and the virtual address range for that particular EPC so in some sense this EPCM is somewhat similar to the page tables which are generally used in systems the only difference is that the most of the page tables are managed by the operating system while with the SGX the EPCM contents is completely managed by the hardware so if we actually go back to this slide and see that there is now a conversion from the virtual address space for the enclave to the corresponding physical address space in the PRM.

Now as will see later there are actually two translation that are involved one is by the standard memory management unit and the page directory and page tables which are maintained by the operating system so this region would then would essentially covert the virtual address to the corresponding physical address second set of validity checks is done by the hardware using the EPCM which is present in the PRM. So this would permit a scenario where the operating system is not trusted and the additional checks done by the hardware would ensure that the enclave code and data is kept safe even when the operating system is untrusted.

(Refer Slide Time: 14:48)



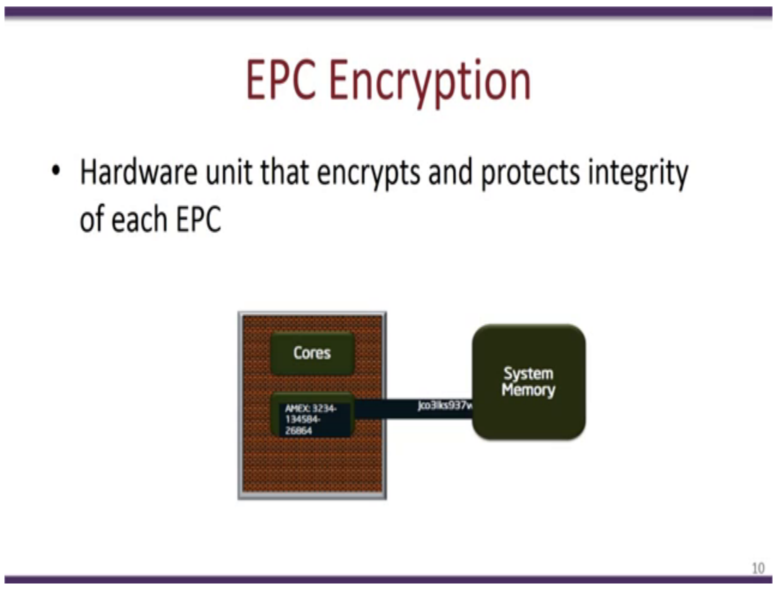
Physical Memory

- SECS: SGX Enclave Control Store
  - One for each enclave
  - 4KB (present in an EPC)
  - Contains global metadata about the enclave
    - EPC pages that are used
      - Mapping information
      - Crypto log of each used EPC page
    - Range of protected addresses used by the enclave
    - 32 / 64 bit operating mode
    - Debug access

9

Now another related data structure which is present in the PRM is known as the SECS or the SGX enclave control store now for each enclave there is one SECS so it is of 4 kilo bytes and what we mean by this is suppose there are let us say 10 processes running in the system and each process have one enclave then there would be 10 SECS structures present in the PRM. So this SECS structure contains global and meta data about that particular enclave, it is used by various reasons to such as to ensure the integrity of the code and data present in the enclave and it is also used for mapping information to the various enclave regions.

(Refer Slide Time: 15:37)



EPC Encryption

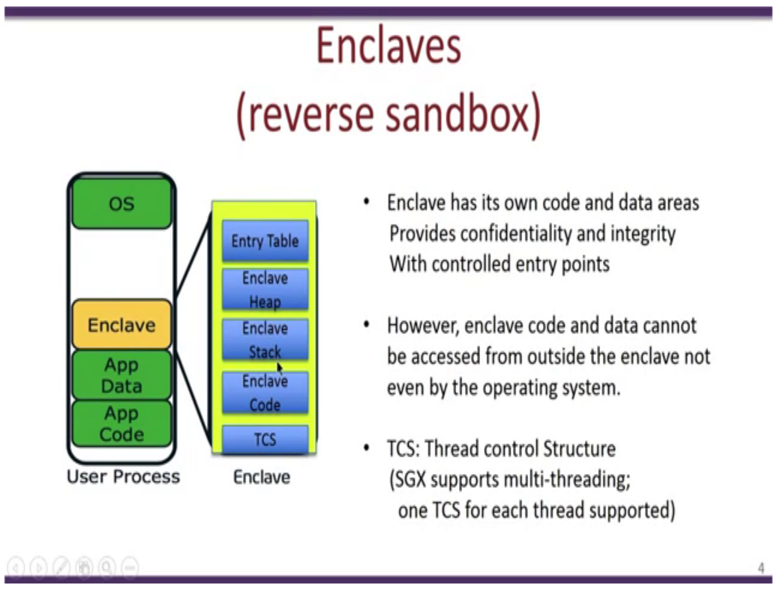
- Hardware unit that encrypts and protects integrity of each EPC

Cores
AMEX: 3234-134584-26864
Jco3lks937w
System Memory
10

So as we mentioned one of the distinctive features of the SGX mechanism compare to the ARM trustzone is that SGX supports encrypted memory so if we look at this so if we consider this as our processor with the multiple codes and so on and we assume that this is a hardware enabled processor then what happens is that any data which is actually sent out of the processor or received from the processor is in an encrypted state when this encrypted data enters into the processor then a very fast decryption engine would decryption it to get the plain text data. So what this means is that any data leaving the processor would be encrypted while any data read into the processor which is present in the enclave would be decrypted.

So this way if we have an attacker who is trying to snoop into the system memory the D-RAM memory for instance or try to snoop into the various buses present in the processor then the attacker would only be able to view the encrypted data so this ensures confidentiality of the enclave region as well as integrity of the data. So for example if an attacker tries to modify this data on this bus checks would be done in the processor to identify that the data has been modified and would throw an exception.

(Refer Slide Time: 17:14)



Enclaves (reverse sandbox)

OS
Enclave
App Data
App Code
User Process
Entry Table
Enclave Heap
Enclave Stack
Enclave Code
TCS
Enclave

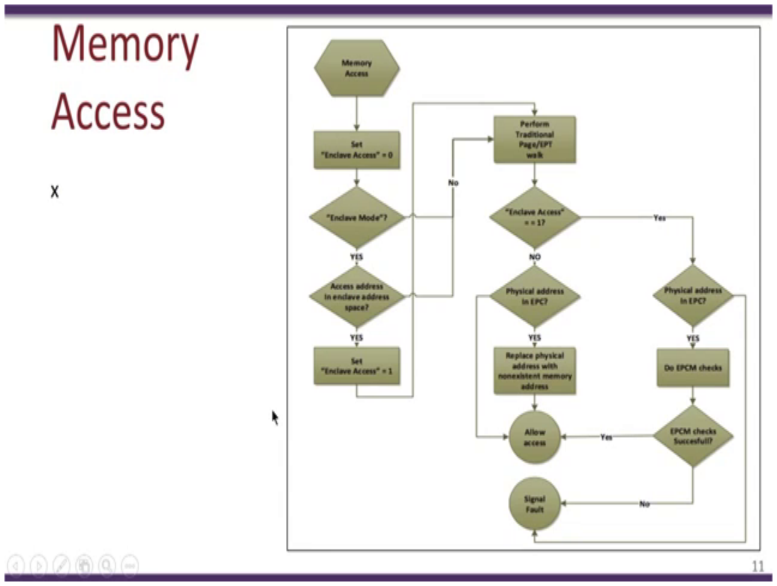
- Enclave has its own code and data areas Provides confidentiality and integrity With controlled entry points
- However, enclave code and data cannot be accessed from outside the enclave not even by the operating system.
- TCS: Thread control Structure (SGX supports multi-threading; one TCS for each thread supported)

4

So recollect the slide where we actually discussed that the enclave region is essentially protected from the various other parts of the process as well as the operating system but however when executing within the enclave region that is you have a function within the enclave region this particular data could access anything in the user space of that particular process. So this leaves us four different alternatives so let us say the first is when you are in the code outside the enclave that is you are executing in normal mode outside the enclave and you are trying to access the data present outside the enclave, so this should be permitted.

The second is when you are executing code outside the enclave but try to access or execute code which is present inside the enclave so this should not be permitted. The two other cases are when you are actually running in the enclave mode and you are trying to access data within the enclave or trying to access data which is outside the enclave so both of this should be permitted by the processor. So let us see how this is managed by the memory management units and the SGX units present in the system.

(Refer Slide Time: 18:37)



Memory Access

X

So we look at this particular flow chart which shows how memory accesses are done in an SGX enclave system. So part of this is done by the operating system and the traditional page tables and page directories which are present the remaining part is done by the hardware especially the SGX aspects of the hardware. So let us consider the case where we have code present outside the enclave and it is making a memory access to data which is also present outside the enclave. So we will follow the flowchart and see that this should be permitted and only the traditional page tables and page mechanisms would work. So we have a memory access that is which is initiated we set the enclave access to zero we check whether it is an enclave mode.

In this particular case it is no so we skip this steps and we go to this place then we perform a traditional page walk and page table, page walk using the page directories and page tables and therefore we will be able to convert the virtual address of that data to the corresponding physical address. Now the next step is this a enclave access so since the data is outside the enclave so therefore no then we check whether the physical address is in the EPC in this case since the data is not in the enclave the data is outside the enclave therefore this case is too is also no and finally we allow the access. So let us see another scenario where you have code present outside the enclave trying to access data which is present inside the enclave.

So will follow this again will set enclave access to zero then is it in enclave mode so it is no so we go here perform the traditional page directly page table walk and obtain the corresponding physical address and check whether it is an enclave access.so enclave access is set still set to zero so it is no again and then we check whether the physical address is in the EPC in this case it is yes and what we do is that we replace the physical address with some non-existent memory address essentially we are going to actually fill in some garbage and permit the access. So what is going to happen over here is that such a memory access would complete but what the program would see is some garbage value and not the actual value corresponding to that memory location present inside the enclave.

We will also look at third aspect where you are running in the enclave mode and trying to access data which is outside the enclave. So will follow this again we set the enclave access to zero as usual we are running in enclave mode so this is yes the access to address in enclave address space which is no so we come here we perform the traditional page table walk and obtain the corresponding physical address then we check whether it is an enclave access equal to 1, no so because enclave access is still zero so we come here and we see that there is a check for physical address in EPC in this case is no because we are in the enclave mode but trying to access data which is outside the enclave and therefore the access is permitted.

Will also look at the final case where we have a code present in the enclave that is you are running in the enclave mode and you are trying to access data which is also in the enclave. So as usual we set the enclave access to zero the enclave mode is yes so we actually come here is the access to address in the enclave address space this is yes so set enclave access to 1 we go here perform the traditional page table walk thing which will actually convert the virtual address to the corresponding physical address then we look at the enclave access equal to 1. So in this case the enclave access is indeed 1 so we come to this part of the flow and check a whether the physical address is in the EPC yes.

Now over here we is the part where we actually look into the EPCM check the various permissions and so on and ensure that this particular EPC where is going to be accessed has indeed the right permissions to permit that particular access. If these permissions are successful then you allow this particular access else we will create a signal fault. In this video we had a brief introduction to Intel SGX in the next video will look at how a move from a software perspective and see how applications are written how the SGX mode is used and how data can be transferred from a normal mode outside the enclave into the enclave and get the result and so on, thank you.