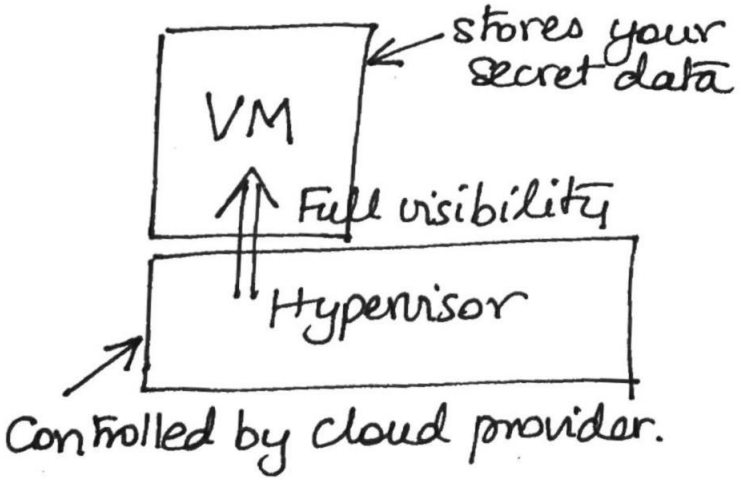
SGX AND ENCLAVES

- Imagine you are a client of a cloud platform. You have some sensitive data (e.g. trade secrets) that you do not want to give to anyone.
- If you take your data to the cloud (e.g., to run data analytics on it), the cloud provider will put it in a virtual machine or a container and run computations on it.
- The cloud provider's infrastructure has a complete view of everything going on in the VM or in the container.



- You are implicitly therefore trusting the cloud provider.
- What about:
  - → malicious cloud administrators?
  - → Government subpoenas on cloud provider?

- How can we solve this problem? Is there any way for client to keep its data private?
- If the client simply encrypts the data, that is not enough.

$$M \Longrightarrow E_{key}(M) = C$$

- Client wants to run some analytics on the data, say $F(M)$. Must decrypt on the cloud ⇒ cloud provider obtains access to data in the clear.
- One solution : HOMOMORPHIC ENCRYPTION.
  A special kind of encryption function $\mathbb{E}_{key}$ such that

$$F\left(\mathbb{E}_{key}(M)\right) = \mathbb{E}_{key}\left(F(M)\right)$$

  So, just decrypt after applying $F(\cdot)$ to $\mathbb{E}_{key}(M)$ and you're done.
- But, still terribly slow. First fully-general homomorphic scheme applicable to arbitrary $F(\cdot)$ functions only invented in 2010 (Craig Gentry, won the ACM PhD dissertation award for this).

ANOTHER APPLICATION SETTING

- Consider two mutually distrusting parties who each have data and want to run some analytics on their combined data.
- Example: Hospital has patient data H

  Insurance company has some insurance pricing models I

  Together, they want to compute some joint function $G(H, I)$
- PROBLEM: Hospital can't just give H to insurance company. Insurance company can't just give data to hospital.
- Solutions? ① Homomorphic Encryption (slowest)
  ② Multiparty Computation (MPC)
  ⇒ Prof. Arpita Patra works on this.
  ③ Use a trusted third party (TTP)

TTP: Both Hospital and Insurance Co give H & I to TTP. TTP runs $G(H, I)$ in a "clean room" and only gives results to both.

But, how to find such a TTP?

THE SOLUTION TODAY : SGX

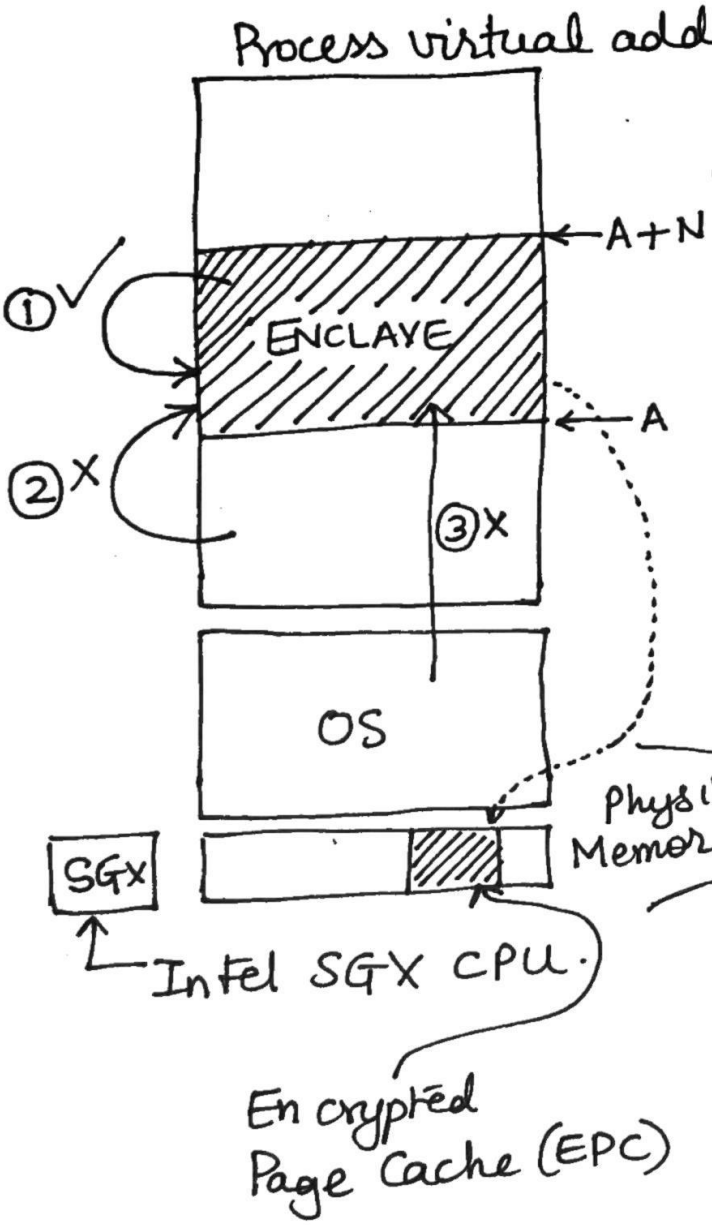
- SGX — Secure Guard Extensions — is a new hardware primitive from Intel
- Available since August 2016 in Skylake series chipsets onwards.
- A kind of trusted hardware that allows clients to protect their data on the cloud. The data is protected even from the most privileged software running on the system, e.g. the cloud provider's hypervisor or the operating system.
- So, what is the SGX?

Here is the end-user's view:

→ SGX introduces new hardware instructions in the hardware's ISA that allows the creation of an enclave in the process address space

→ Data stored in the enclave is not accessible in plain text form from the rest of the process or even the operating system.

→ Code can also be placed in the enclave and can operate on the data stored in the enclave. The code in the enclave has plaintext access to data stored in the enclave.



An enclave:

- is a linear region of a process virtual address space
- virtual pages of enclave are mapped to a specific region of memory called encrypted page cache.
- the SGX CPU encrypts all contents of data stored in enclaves with a key known only to hardware.
- Access ① allowed
- Access ② & ③ disallowed (i.e., plaintext data will not be visible).

If you had this primitive, you can solve both our motivating problems.

- You put your data in an enclave on the cloud provider. Put the code for $F(\cdot)$ also there. Set up a TLS to the enclave. get results. cloud provider wont learn anything.
- Hospital and Insurance company identify a cloud provider who offers enclaves:
  - → Hospital securely loads H into enclave
  - → Insurance provider loads I.
  - → Either one loads G. The other verifies it is loaded correctly.
  - → output $G(H, I)$ given to both.

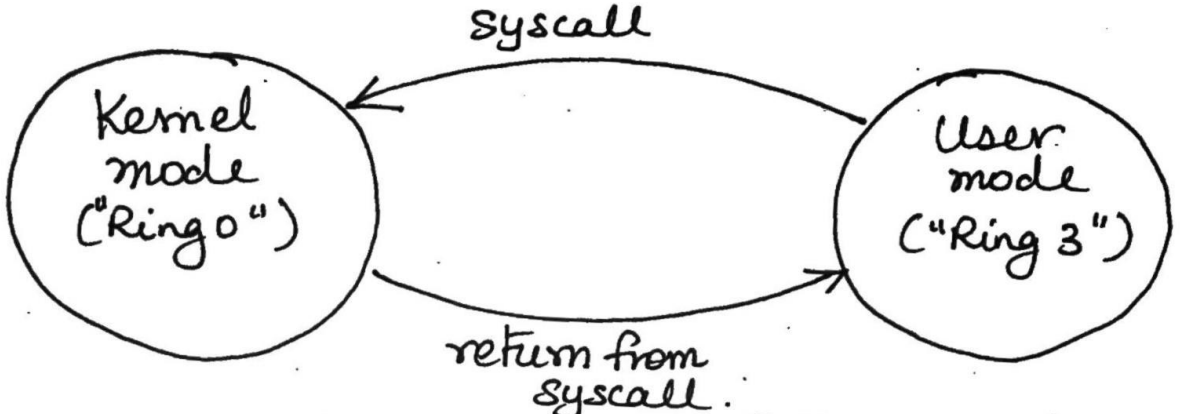
Cloud provider need not be mutually trusted.

OPEN: How to ensure data (e.g. H, I) and code (e.g., F, G) loaded correctly?

That is called provisioning the enclave. We will discuss that at the end.

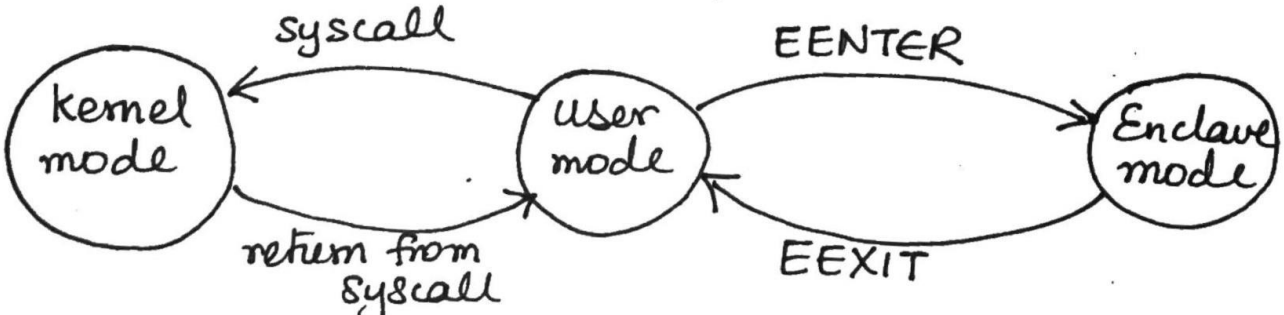
HINT: ATTESTATION PROTOCOLS! 

- But first, how does SGX enable this magic?
- On a "normal" CPU, two processor modes



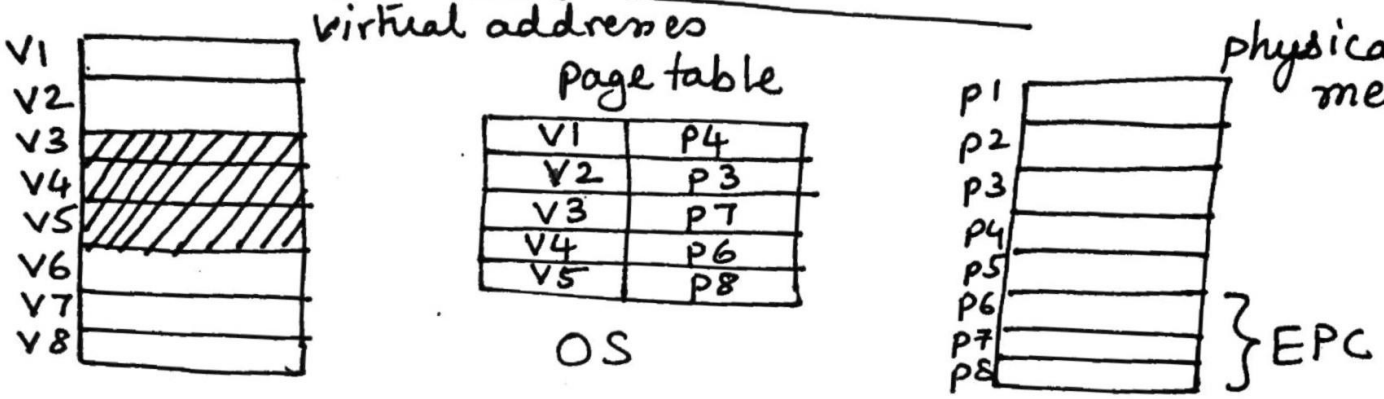
Kernel mode is all powerful. Full acess to the virtual address space of the process.

- On an SGX CPU



- Cannot go directly from kernel mode to enclave mode and vice-versa.
- Kernel mode & user mode do not have plaintext acess to virtual addres space of enclave.

IMPLEMENTATION OF Enclave mode



- Hardware ensures that enclave virtual pages are only mapped to EPC region of physical memory (memory controller ensures this). Note that page table still controlled by the untrusted OS. OS could even be adversarial, but it does not matter. Hardware ensures the mapping goes only to EPC

- A memory access happening to V3, V4 or V5.
  - → Processor checks: is processor in enclave mode?
  - → Ans: no: → present encrypted view.
  - → Ans: yes → Bring data from EPC to the cache line (in CPU package) and decrypt the data.
  - → on eviction of cache line, re-encrypt & store in EPC.

Thus:

→ kernel code cannot access enclave

→ process itself cannot acess enclave.

→ to access enclave, must execute EENTER, which takes you to code executing within the enclave!

→ So only enclave code can access enclave data.

- OK, so, we've solved the problem of secure data access on an untrusted cloud provider.
- But we haven't solved the bootstrap problem yet, i.e, how do we get the data into the enclave in the first place?
- Can we encrypt the data and send it to the enclave?
  - Possibly. But how will the enclave decrypt it?
  - You've to get the key to the enclave first how?
  - Can you & the enclave do Diffie-Hellman key exchange? But how do you certify the public key?

- The answer boils down to trusted hardware. SGX also provides TPM-like functionality.
- The hardware has a public/private key pair. (This key pair is different from the one used to encrypt enclave contents. Those are ephemeral hardware generated keys).
- The public key of an SGX chip is digitally certified by Intel CA. The private key is "burned" into the silicon itself.
- When there is a request to create a new enclave (ECREATE instruction) of a particular size, the hardware creates an "empty enclave" with some bare-minimum code to do crypto operations.
- A public/private key pair is generated for the enclave ★ Pub_{enc} / $Priv_{enc}$.
- $Priv_{enc}$ is stored in the enclave, and Pub_{enc} is made part of the "enclave creation report"

- The client can request an attestation report that the enclave was created properly.
- The SGX hardware provides an attestation report digitally signed by the hardware's private key to the client.
- Attestation is conceptually similar to TPM-based attestation, but the low-level details slightly differ (e.g., no PCR registers, no tpm_extend, etc., but hardware has a way of "measuring" the newly created enclave).
- This attestation also therefore attests the enclave's public key Pub_{enc}, which is therefore indirectly certified by Intel's CA.

Thus, now client can engage in TLS handshake with the enclave & send its code/data to the enclave! Everything is encrypted, from client's end to inside enclave.

Data decrypted in enclave, but invisible to -cloud provider anyway!

SGX supports more features

- "Sealing" – binding data to a particular physical computer.

Shortcomings:

- Porting code is a problem. Enclave does not allow system calls directly from within.
- System call return values become an attack vector (solution in next class – SCONE)
- Side channel attacks – page access pattern is visible to underlying OS.

Summary:

Big leap forward in enabling secure cloud computing

→ Cloud provider untrusted

→ No access to confidential client data

→ Yet client can compute on that data.

Next big advance: Intel TDX (Trusted Domain Extensions). Look out for it soon!