TRUST

- Trust unlike in other applications is considered to be a bad thing in Computer Security. Trust=Reliance=Dependence
- If you trust something/one then the misbehavior of that thing or person ay impact your security goals.
  
  Considering the example of amazon.com, let us show diagrammatically what all things it has to trust:

AMAZON

- Example of Transitive Trust (i.e. if we depend on ‘X’ and ‘X’ depends on ‘Y’ then we indirectly depend on ‘Y’ (Here ‘X’ and ‘Y’ could be some third parties))

- Example of “TRUSTING TRUST”

This example is of a Linux inventor. Now some backdoor could be inserted in the source code. But this would be evident by looking at the source code directly. So now the bad guy could insert the back door in the compiler. But even this could be found out if someone sees the code of the compiler or uses some kind of a line by line debugger. So now the bad guy could insert this backdoor code while compiling the compiler and this could go on further.
- **How could we fight this?**
  1. Limit the no. of different compilers (somehow thoroughly check them)
  2. Build the compiler yourself
  3. Introduce transparency of components
  4. Redundancy (Do not depend on solely one person for any kind of equipment or work. Create redundancy so that there is no 100% trusting on any one)
  5. Debug the program interaction
  6. Limit the access of developers
  7. Audit the developers
  8. Automation

- **TRUSTED COMPUTING BASE (TCB)**
  This is a set of s/w and h/w which you trust to achieve your security goals. **Advise: Keep the TCB as small as possible.**

- **Hardware Trust**

  ![CPU diagram]

  Rings of Trust  
  (0: Privilege Mode, 1: In this mode IHA and Clock are not accessible)

**How can the computer support (mutually) distrusting users?**

1. Separate the memory of different users
   - To ensure: **Confidentiality** **Integrity** and **Availability**
2. Add the user mode register to the CPU
3. The code must be running with user mode=0 in order to modify IHA, clock or other interrupts, user mode register, PTR.
   - At boot, user mode=0
   - Before switching to execute an application, kernel sets the user mode to 1
   - Interrupt sets the user mode to 0

4. Kernel maintains page tables for each process.
   - Kernel sets PTR before switching to process
   - Gives each process different physical pages

- **Communication in Applications**
  1. Communicate via shared physical pages
     `runquery(key, userid)`
     ```
     if(owner(key)=userid)
     retval(key)
     ```
     Time of check to Time of User: race condition

**How to enable access control for application communication?**

(To be covered in the next lecture)