Chip Multi Processors, System On a Chip, and Multi-socket Protections

Chapter 7

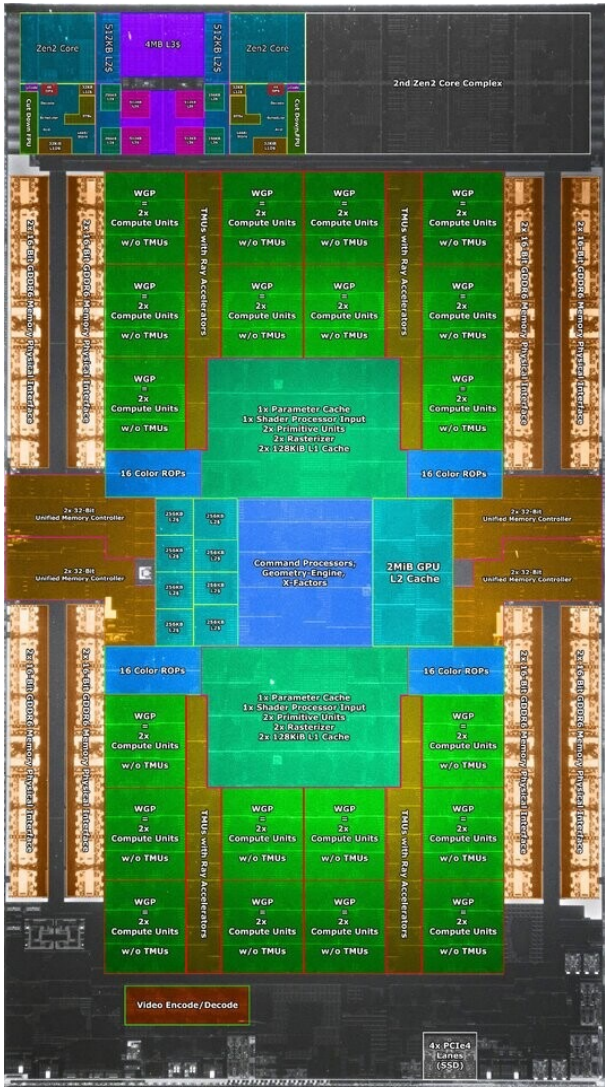
[1] J. Szefer, “Principles of secure processor architecture design,” *Synth. Lect. Comput. Archit.*, vol. 13, no. 3, pp. 1–173, 2018.

Security Challenges on Multiprocessors

- Multi-socket Multiprocessors required off-chip communication between cores
  - Susceptible to similar attacks of memory (e.g., probing, physical interchange with rogue chips, …)
  - Require solutions (in the processor design) to guarantee system **confidentiality**, **integrity** and **authenticity**
- Chip Multiprocessors (CMP)
  - Threats move inside the chip
    - Less susceptible but (in Systems-on-a-chip or SoC) might include many IPs (accelerators, ASICs,..)
    - Certain IP can be malicious

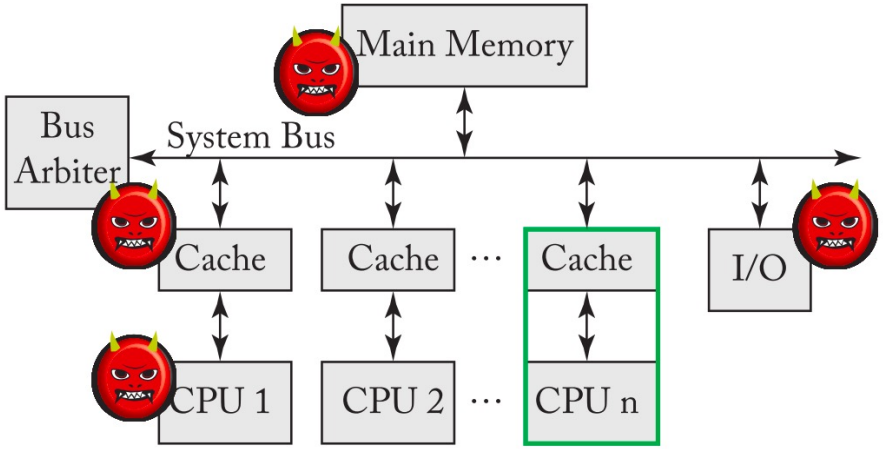
Aside: SoC

- Solution to Dark silicon: use specialized hardware for key operations

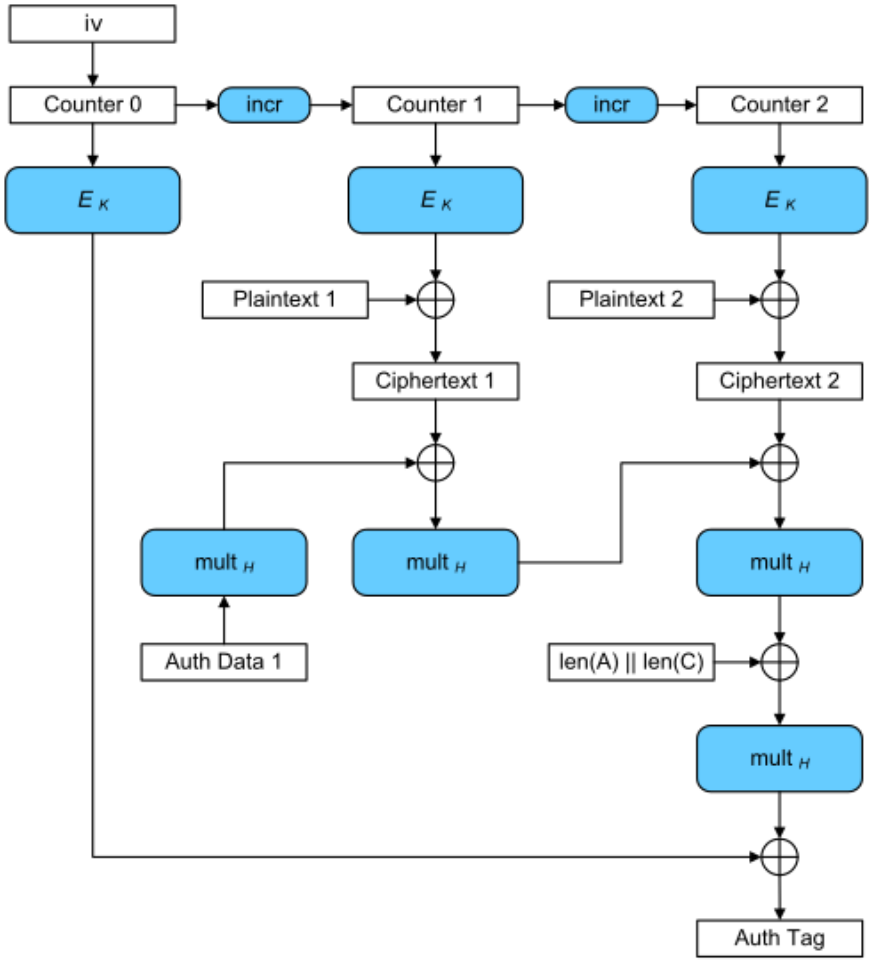


UMA Threat Model: Communications

- Confidentiality and Integrity
  - Communications (assumes pre-shared keys)
    - Counter mode AES (pregen. a counter encryption and *xor* with actual data)
    - Challenge to track the counters (pair origin destiny → higher storage reque., shared counter → increases protocol complexity)
    - For integrity use MAC
      - Can be combined: AES Galois Counter Mode (AES GCM)



Aside: AES GCM



UMA Threat Model: Memory

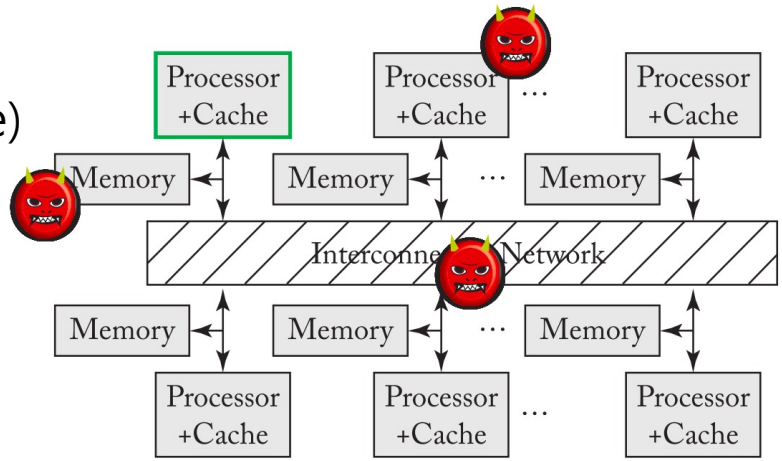
- Confidentiality and Integrity
  - Use Merkle trees for integrity
  - Single Tree
    - Processor's "share" the memory: they has to reach a consensus about the root tree value (which is some processor)
      - Bus simplifies the problem: snooping can allow to authenticate messages and update the root accordingly
  - Multiple Tree
    - Each processor handles a tree of the data is working with (data privately handled by other processors appears as null leafs)
    - Exclude shared memory from integrity checks (not in any tree)
      - Copy it in regions where integrity is performed
- Pattern Access protection
  - Somewhat easier to do with SMT (if assumed malicious). Not needed physical probings

UMA Thread Model: Key Management

- Each processor has to have its own K_r
  - Other processors should be informed of the legitimacy of such keys
  - But they to use a common key encrypt comms. and memory
- Generate a key at boot time a shared key? (pre-install)
  - Susceptible of being attacked at boot time
- Public key cryptography
  - Can be too expensive (in hardware terms) just for sharing keys in a SMP systems
  - Doing it at system first boot or when a processor is added to the system?
  - A malicious attacker can insert a rogue processor just to steal the key

NUMA Threat Model

- None of the processors have a global vision of what is happening in the system
- Confidentiality and Integrity
  - Processor-to-processor comms AES or AES GCM
  - Need to be accommodate within the coherency protocol (prohibitive to cypher and decipher all messages)
    - Coherence protocol can check if messages "are" from the normal path (i.e., detect if a rogue processor is not trying to stole a cache block)
  - AES might be too Slow ?(0.25B per cycle) in a 25 GB/s network (QPI)
    - Try to encrypt only "important" things?
    - Faster ciphers?

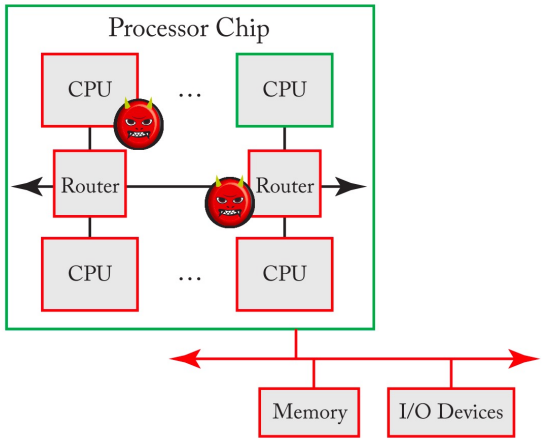


NUMA Threat Model (cont…)

- Access Pattern Protection
  - Easier to obfuscate than UMA (not all processors see all memory requests)
- Key Management
  - Use pair-wise communications to interchange keys
  - Prevent attacker to extract the keys using a malicious processor
  - Keys needs to be protected from read by any untrusted software (potentially untrusted OS or hypervisor)

Threat Model for CMP + SoC (MPSoCs)

- Usually, many IP inside a single chip: increased risk of having a malicious component (both in the supply chain and manufacture process)
- Processors or Accelerators can tamper with memory
- Routers can tamper packets (i.e., tamper memory)
- NoC wires are more easily detectable to external probing (i.e., packets can be accessed or modified)



Communication Protection Mechanisms

- Packets are broken down into flits->phits
- Phits should move fast (1-cycle from router to router): no room for encryption
  - Optimized (big) AES 4bytes/cycle, lightweight cyphers 1 byte/cycle while phits are 4bytes
- Solutions
  - Combine network coding into the cyphering strategy
  - Injection time
- Memory encryption integrity is easier in single socket systems

3D Integration Considerations

- Higher integration make less easy to access to the wires
  - Might need to change the threat model