Challenges & Approaches for Future Secure Execution Environment Design

Yan Solihin

Cybersecurity & Privacy Cluster

Professor & IEEE Fellow

UCF

Yan.solihin@ucf.edu

Outline of The Talk

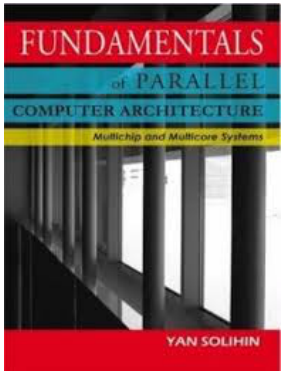
- A little bit about myself
- Motivation
- Secure Execution Environment Milestones
- Future Challenges
- Conclusions

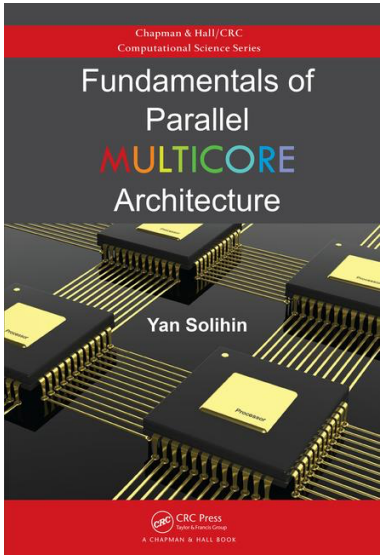
Outline of The Talk

- A little bit about myself
- Motivation
- Secure Execution Environment Milestones
- Future Challenges
- Conclusions

A Little Bit About Myself

- 2002-2018: Professor of ECE, NCSU
- 2008: GCOE Visiting Prof, Waseda Univ, Japan
- 2011: Consultant, Intellectual Ventures
- 2016-2018: Program Director, NSF
  - SaTC, XPS/SPX, CSR, BigData
  - Co-Founded NSF/Intel Partnership on Foundational Microarchitecture Research (FoMR)
- 2018-present: UCF
  - Director of Cybersecurity & Privacy Cluster, Prof in CS
- ARPERS research group
  - 14 PhD grads + 7 PhD students (2 at NCSU + 5 at UCF)





UCF Cyber Security & Privacy Cluster

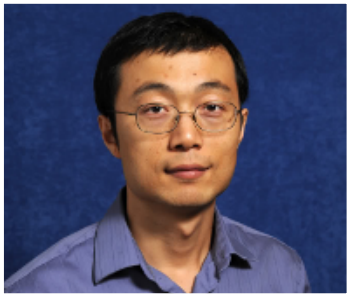
Dr. Yan Solihin, Cluster Lead

Dr. Aziz Mohaisen

Dr. Clay Posey

Dr. Gary Leavens

Dr. Cliff Zou

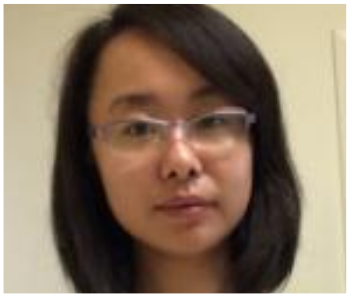


Dr. Paul Gazzillo

Dr. Pam Wisniewski

Dr. Amro Awad

Dr. Yao Li



Research Areas

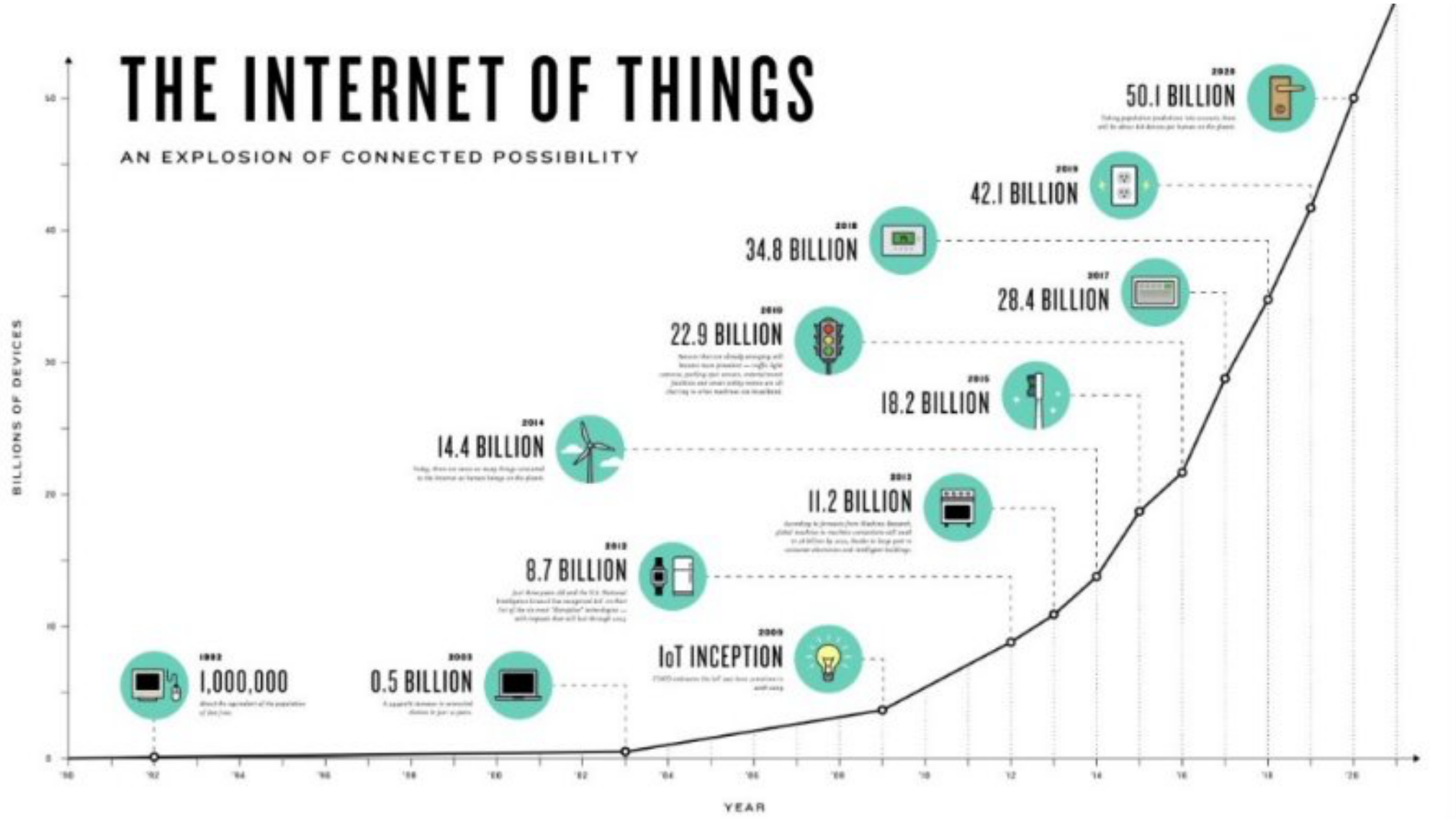
- **Trustworthy Cloud** (secure enclave, side channels): Yan, Gary, Paul, Amro
- **Blockchain** (smart contract, crypto jacking, scalability): Aziz, Paul
- **Secure Machine Learning** (adversarial ML, integrity protection, recovery): Aziz, Yan
- **Organizational and behavioral** (insider threat, policy compliance): Clay
- **Privacy** (networked privacy and online safety, privacy-oriented architecture, data enclave): Pam, Yan, Clay
- **Malware** (analysis): Aziz, Cliff
- **Digital forensics** (fraud detection and forensics): Cliff
- **Software security** (formal methods, software engineering): Gary, Paul
- **IoT se**

We expect to hire 2 tenure-track Assistant Professors for 2020. Please spread the word!

Outline of The Talk

- A little bit about myself
- Motivation
- Secure Execution Environment Milestones
- Future Challenges
- Conclusions

Internet of Things



Source: internet

Exponential Growth of Data

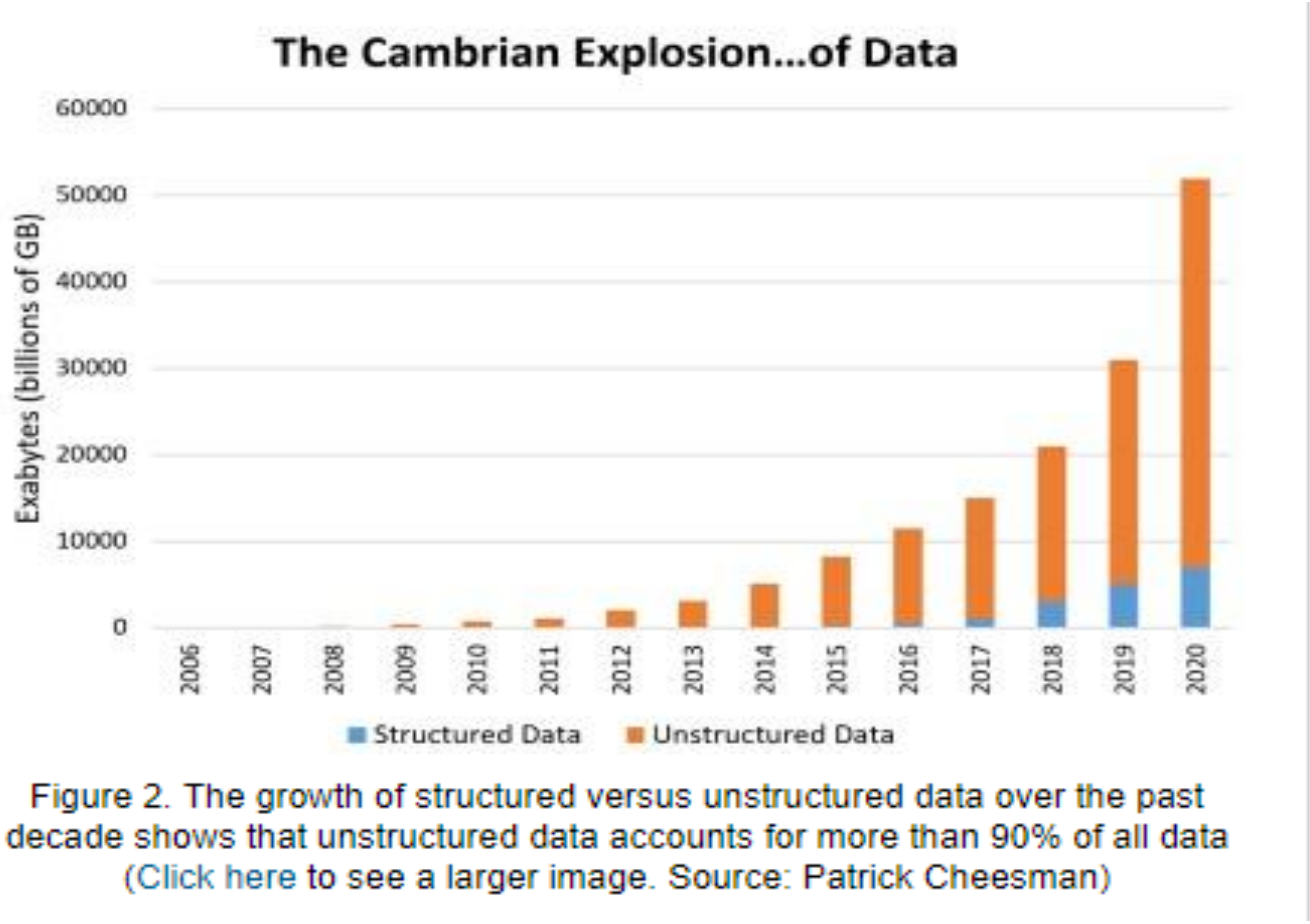
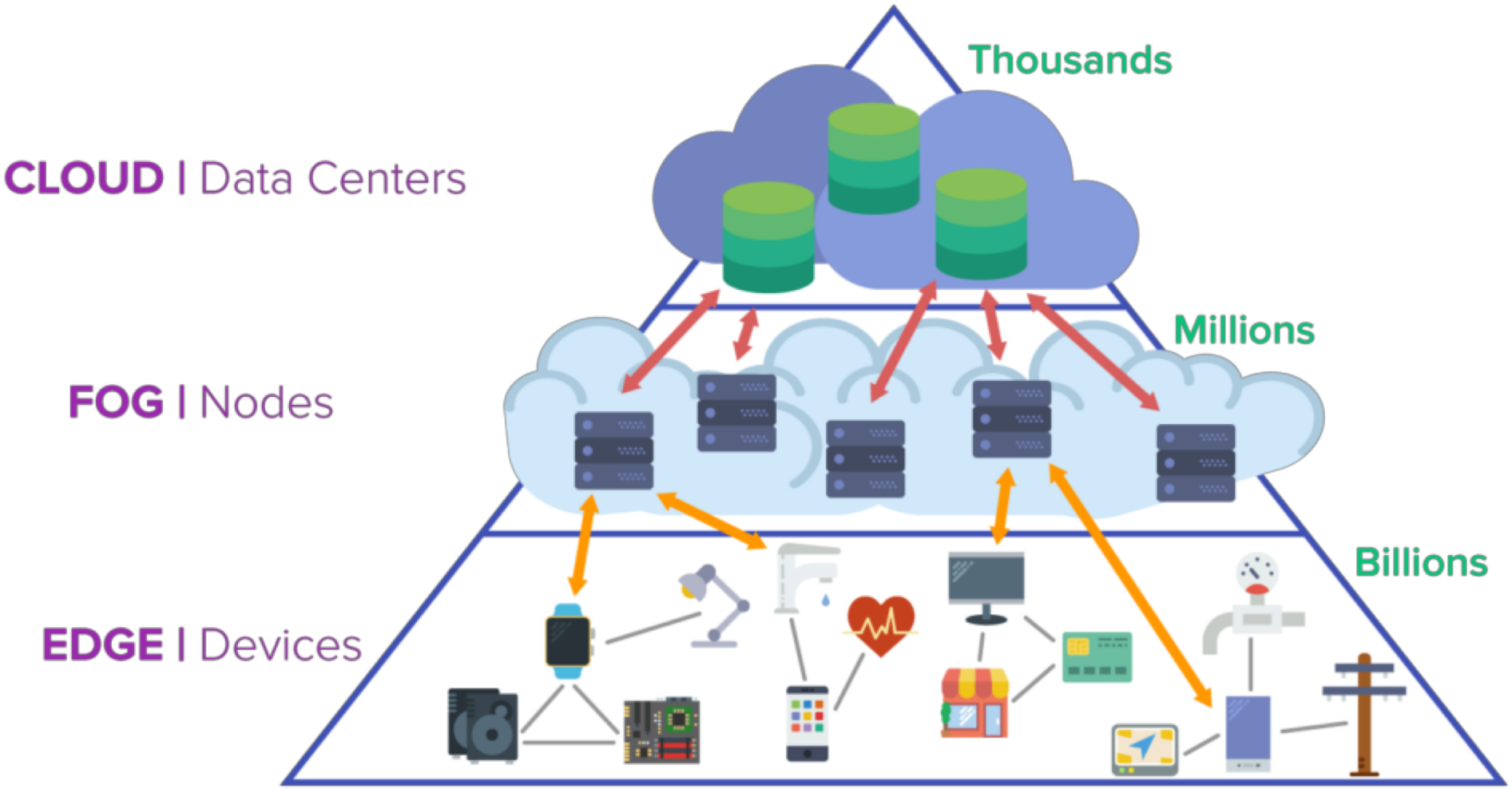


Figure 2. The growth of structured versus unstructured data over the past decade shows that unstructured data accounts for more than 90% of all data (Click here to see a larger image. Source: Patrick Cheesman)

•100 terabytes of data uploaded daily to Facebook. (Source)

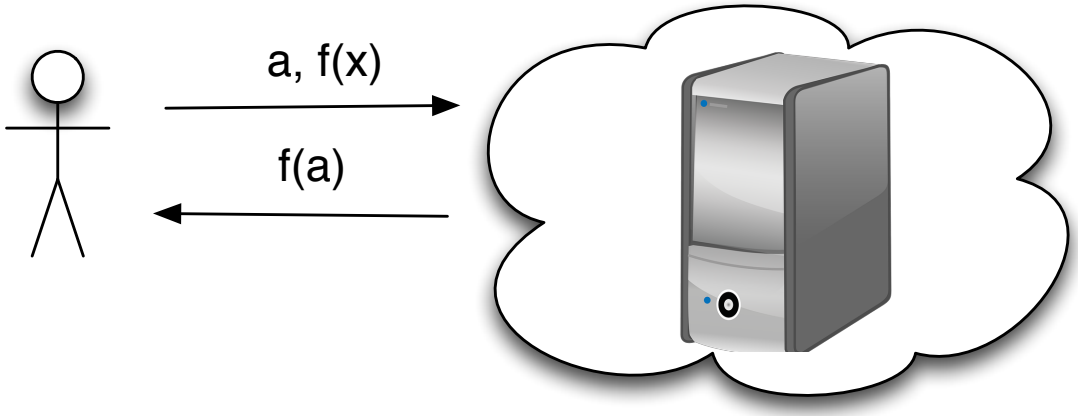
•YouTube users upload 48 hours of new video every minute of the day. (Source)

Cloud and Edge Computing Model



Source: internet

Secure Execution Environment (SEE)

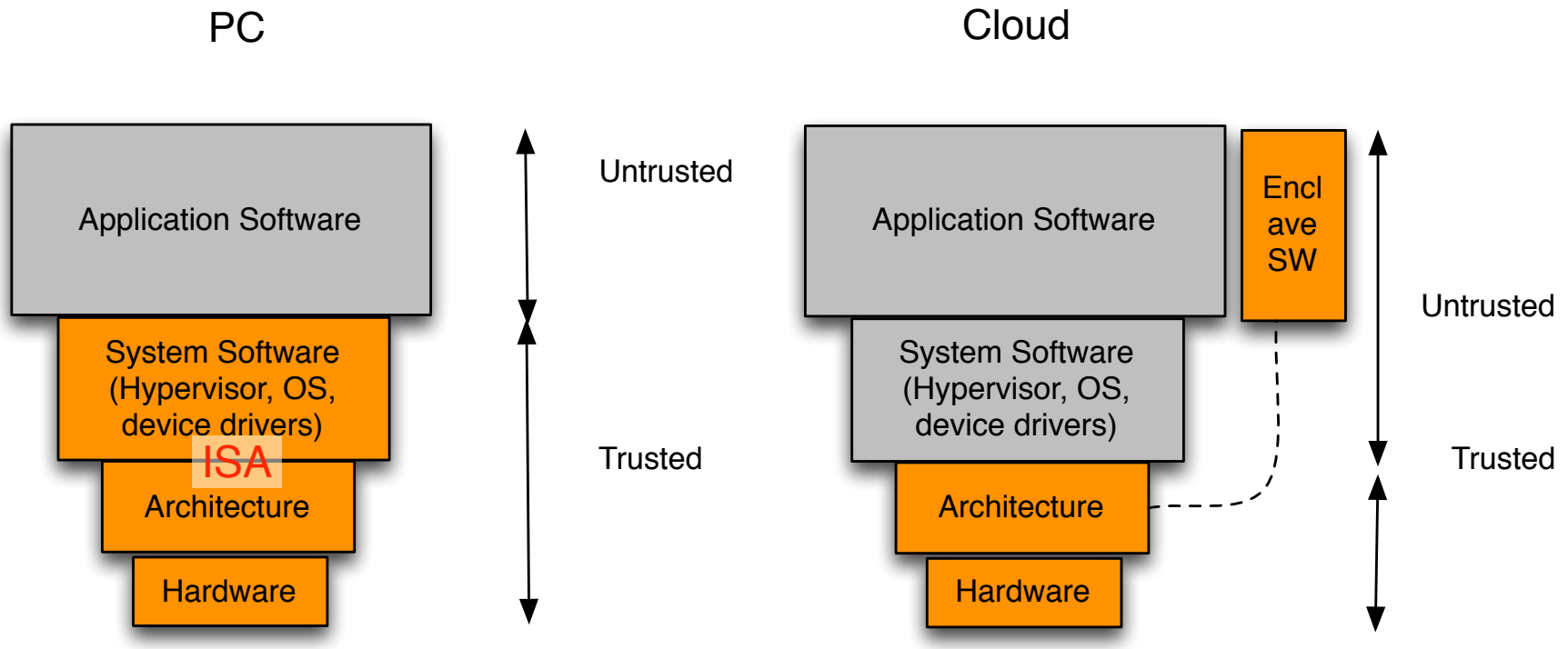


- Goal:
  - Confidentiality of a, f(x), and f(a)
  - Integrity of f(x)
  - User receives f(a)
- What could go wrong?

Threats

- Data centers warehoused in a large facility
  - Physical security usually very good
- Business model requires high utilization, through virtualization and hardware sharing
- Vulnerabilities
  - Large attack surface
    - System software (hypervisor and OS) security point of failure
    - Hypervisor (200-800K LoCs) and OS (e.g. 50M LoCs)
  - Side channels threats: attacker VM may infer secret from behavior of victim VM
  - SEE attempts to reduce the Root of Trust

From PC to Cloud



- System SW is trusted
  - Windows XP ~50M LoC
  - Device drivers, Hypervisor, ROM, microcode, etc.
- Security is software responsibility

- TCB only HW+Arch
  - System SW untrusted
- Arch provides secure environment (e.g. enclave)

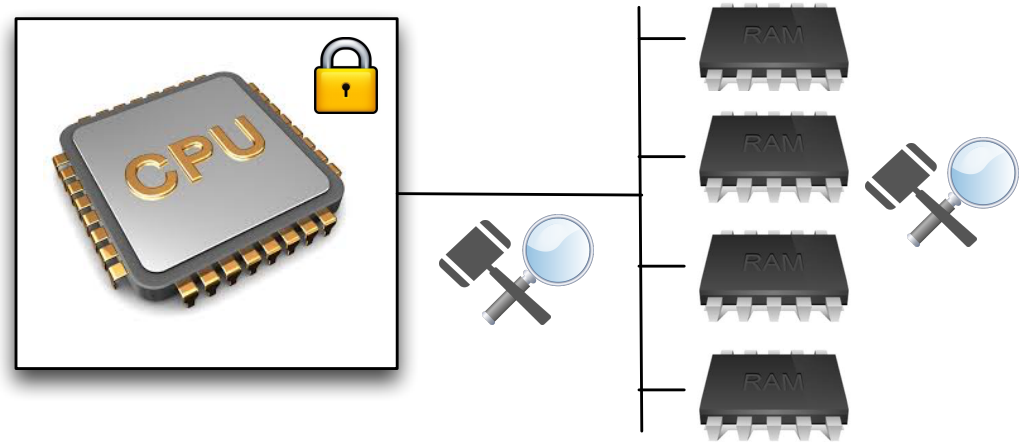
Industry Solution: TEE

- Several competing models

Technology	HW Trusted Base	SW Trusted Base
TPM	Motherboard (CPU, TPM, DRAM, buses)	All software
TrustZone	CPU chip	Secure world (firmware, OS, apps)
Secure Processor (XOM, Aegis, SGX, etc.)	CPU chip	Application + small SW

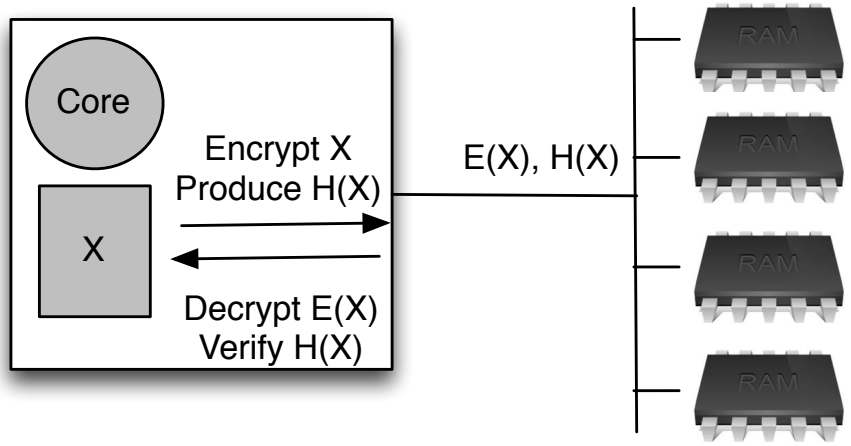
Source: Devadas

A Look Back: Secure Processor



- CPU as the trusted secure parameter
  - Functionally correct and bootstrapped securely
- Everything else untrusted
  - System bus, memory, I/O, prone to snooping/modifications
- Requires memory encryption and integrity verification
  - (also key management, attestation, not part of the talk)

Memory Encryption and Integrity Verification



- Writeback data block X from Cache
  - Encrypt X, compute H(X), then store in memory
- Cache miss on data block X
  - Fetch E(X) and H(X)
  - Decrypt E(X), compute H(X) and compare vs. Mem[H(x)]
  - Verify freshness of X

Outline of The Talk

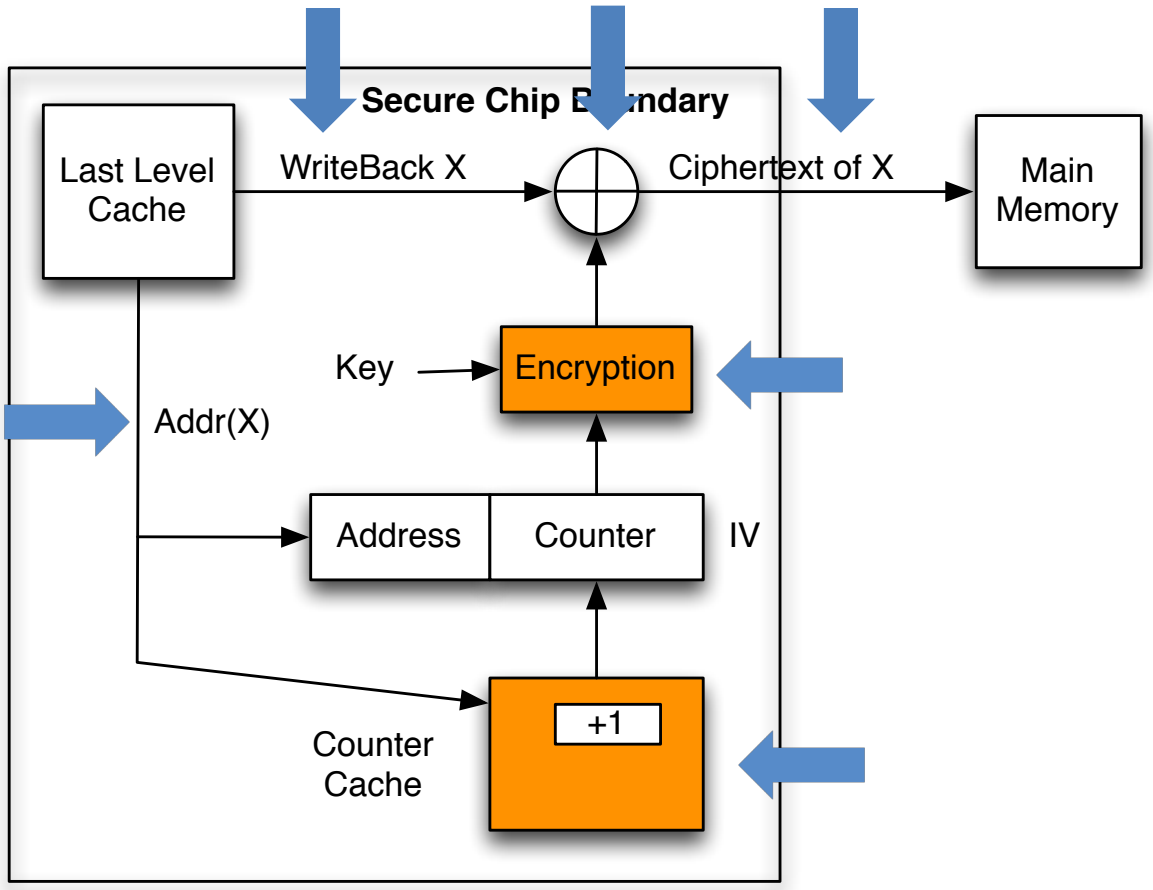
- A little bit about myself
- Motivation
- Secure Execution Environment Milestones
- Future Challenges
- Conclusions

Key Milestones

- XOM (2000): memory encryption
- Yang et al. (2003): counter-mode encryption
  - Encryption delay removed from critical path delay

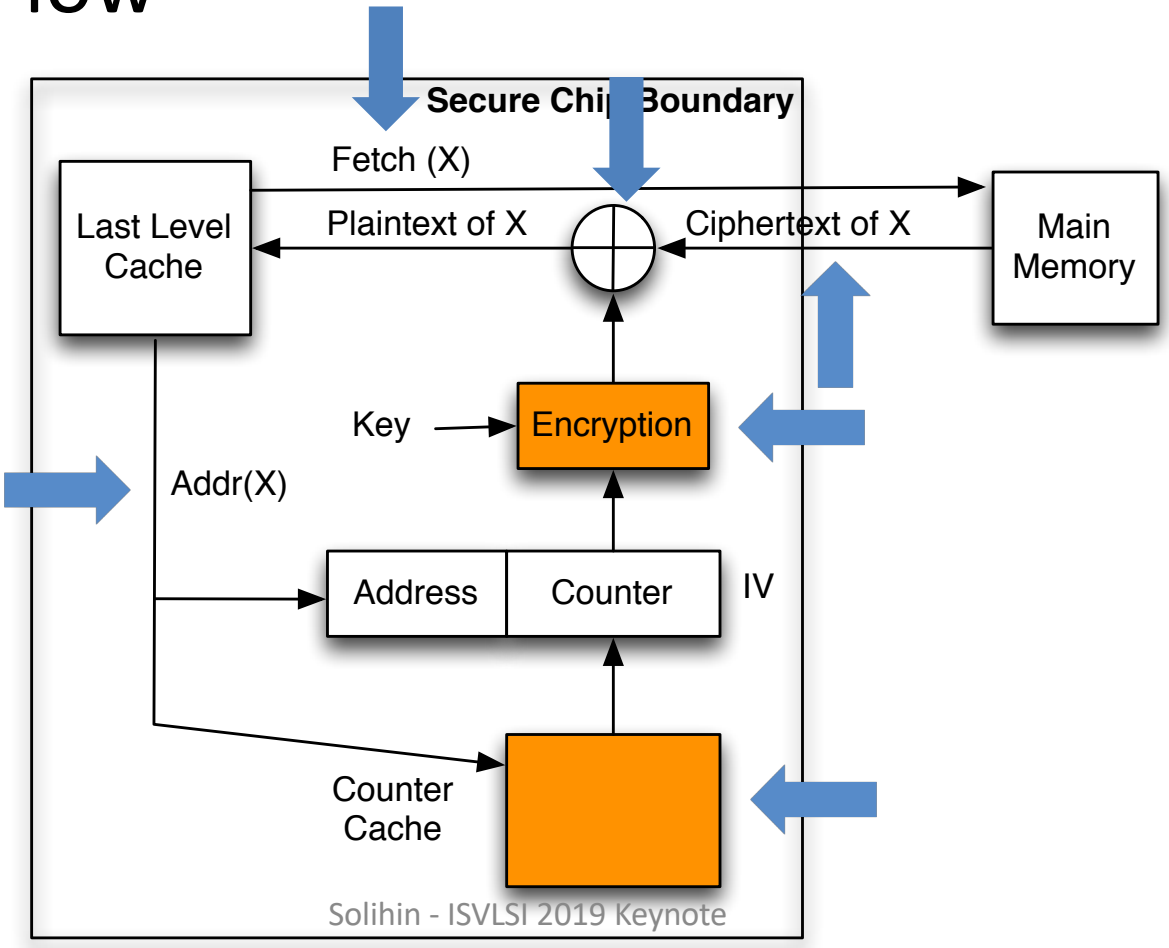
Encryption Process

- Key to security: counter must never be reused



Decryption Process

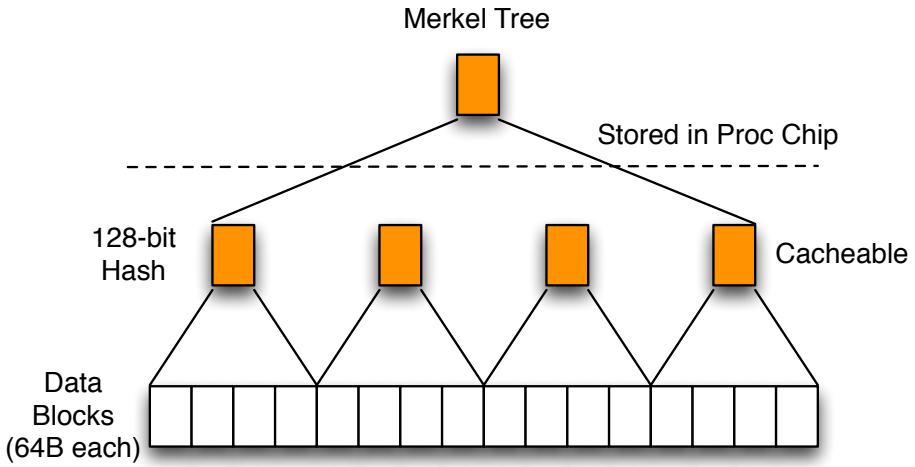
- Key to performance: counter cache miss rate must be low



Key Milestones

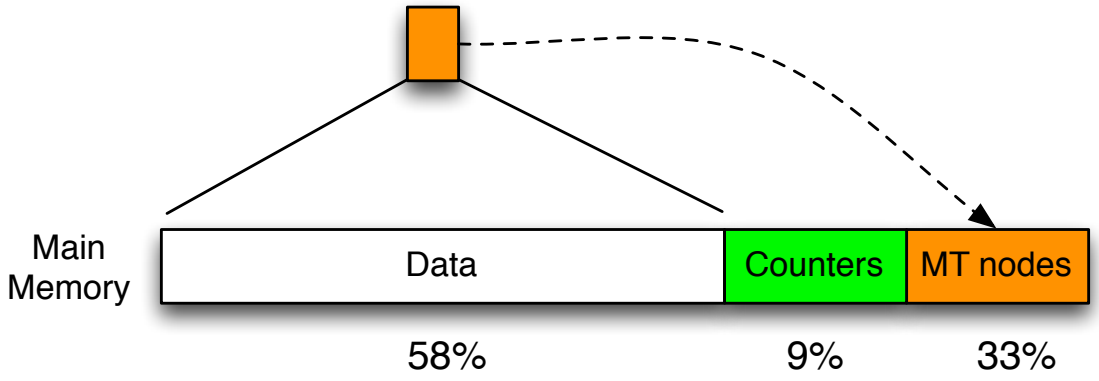
- XOM (2000): memory encryption
- Yang et al. (2003): counter-mode encryption
  - Encryption delay removed from critical path delay
- AEGIS (2003): Merkle Tree memory integrity verification

Merkle Tree



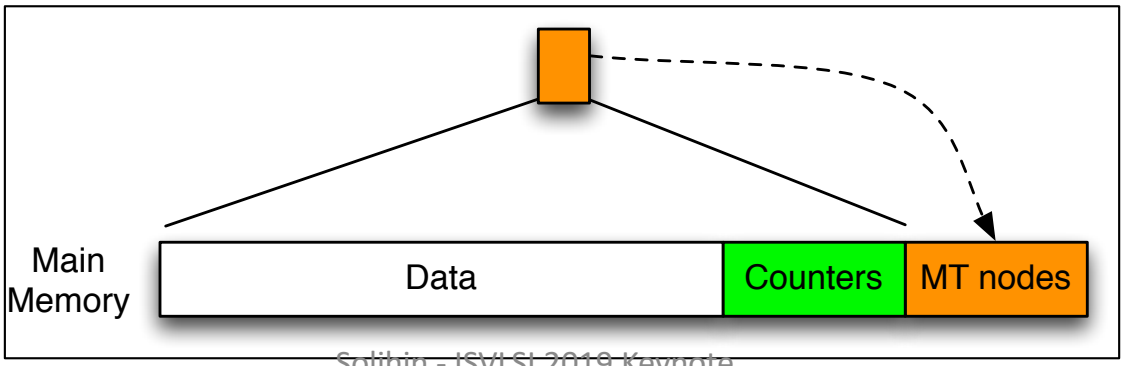
- A tree of hashes covering all data in memory
- Root always kept on chip
- Non-root are cacheable
- Verification by computing hash and comparing it up the tree

- ~Half of main memory unusable for data:
  - MT (33%) + Counters (9%)
- Insecure



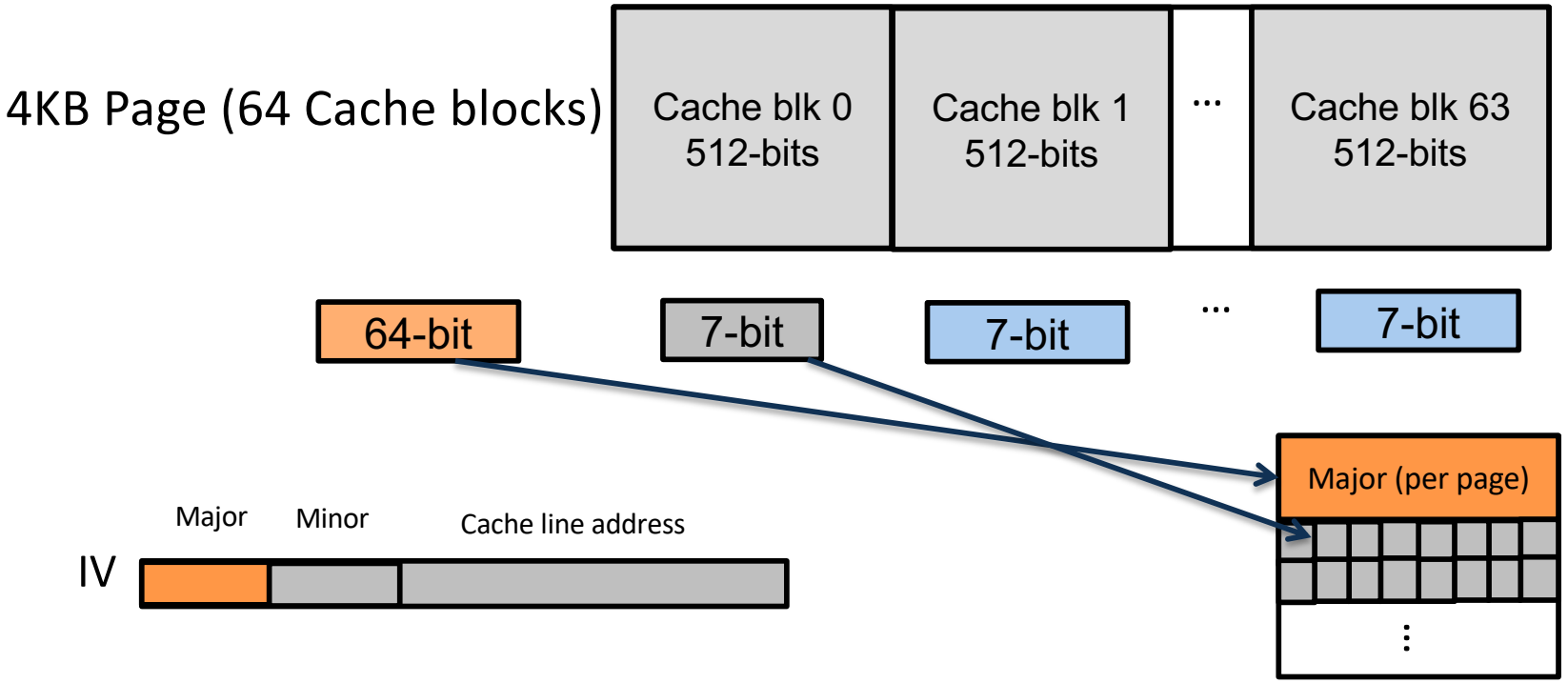
Key Milestones

- XOM (2000): memory encryption
- Yang et al. (2003): counter-mode encryption
  - Encryption delay removed from critical path delay
- AEGIS (2003): Merkle Tree memory integrity verification
- Yan (2006): discovery of counter-rollback attacks,



Split Counter Organization [Yan'06]

- Monolithic counter: too big (high counter cache miss rate) vs. too small (frequent whole memory re-encryptions)
- Use Split Counter instead

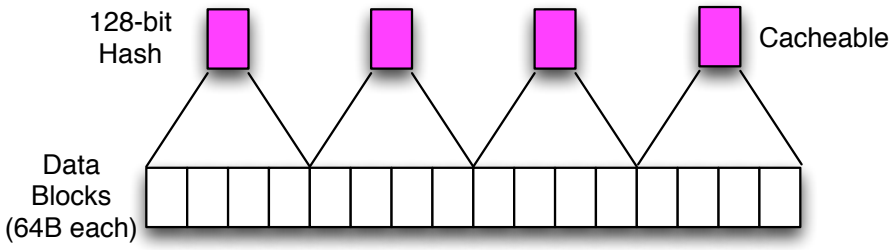


- Major counter never overflows => no whole memory re-encryption
- Effective counter size is 8 bits for a 64B block (88% reduction!)

Key Milestones

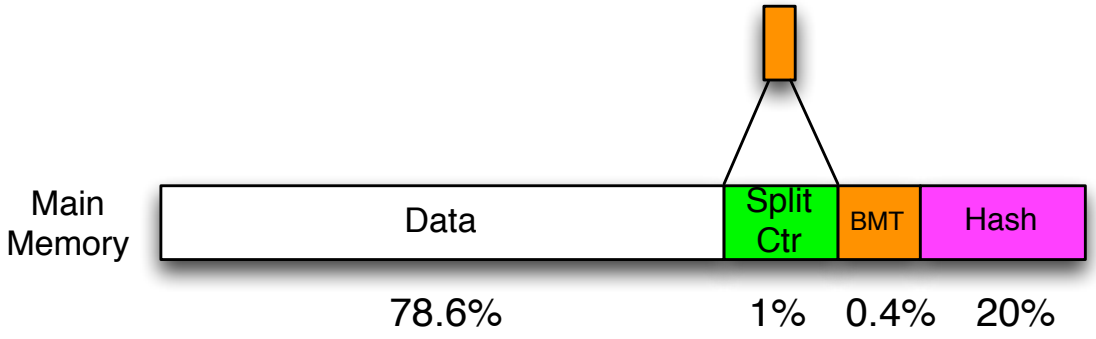
- XOM (2000): memory encryption
- Yang et al. (2003): counter-mode encryption
  - Encryption delay removed from critical path delay
- AEGIS (2003): Merkle Tree memory integrity verification
- Yan (2006): split counter, discovery of counter-rollback attacks
- Rogers (2007): Bonsai Merkle Tree

Bonsai Merkle Tree [MICRO'07]

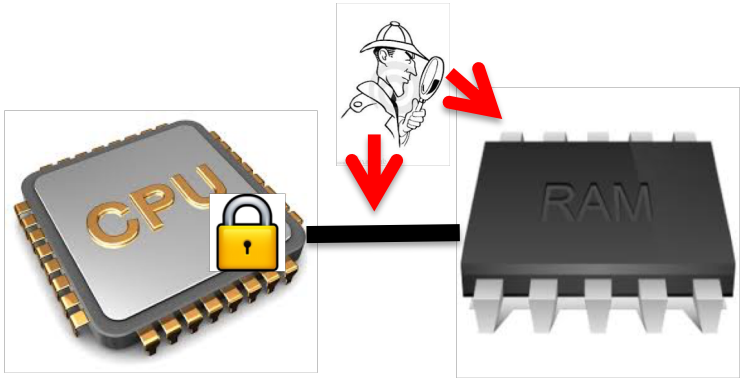


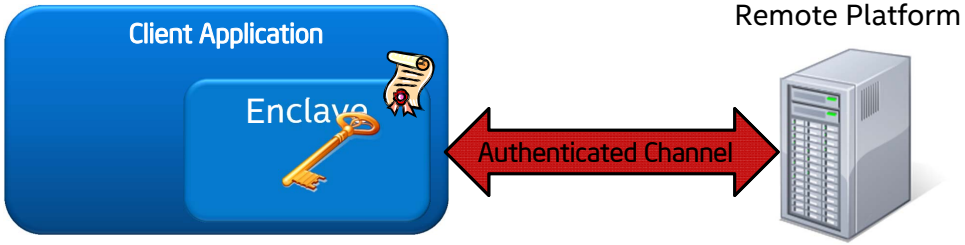
- First to consider integrity protection for counter-mode encryption
- Protect data using *stateful MAC*
- **Key**: only counters need freshness protection
- Bonsai MT = MT over counters only!
  - Same security guarantee

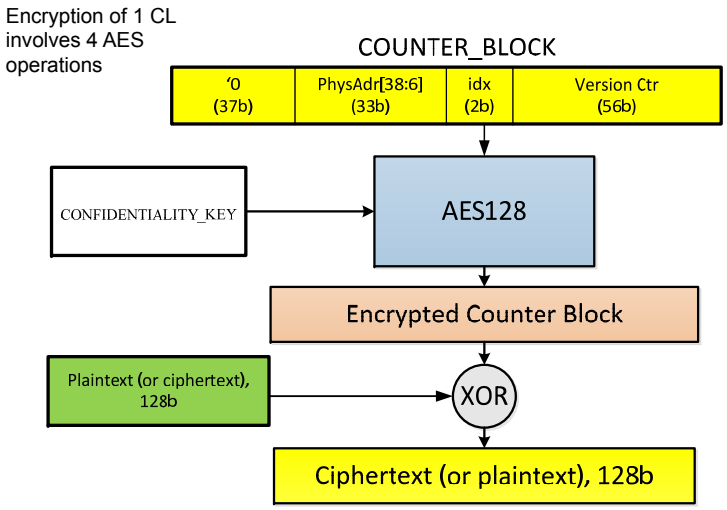
- Counter + BMT only take up 1.4% of main memory
- Total for security 21.4%



Trusted Execution Environment in SGX







Source: Intel

- CPU as the secure parameter
- Attestation of enclave code
- Key sealing
- Memory encryption and integrity verification

Outline of The Talk

- A little bit about myself
- Motivation
- Secure Execution Environment Milestones
- Future Challenges
- Conclusions

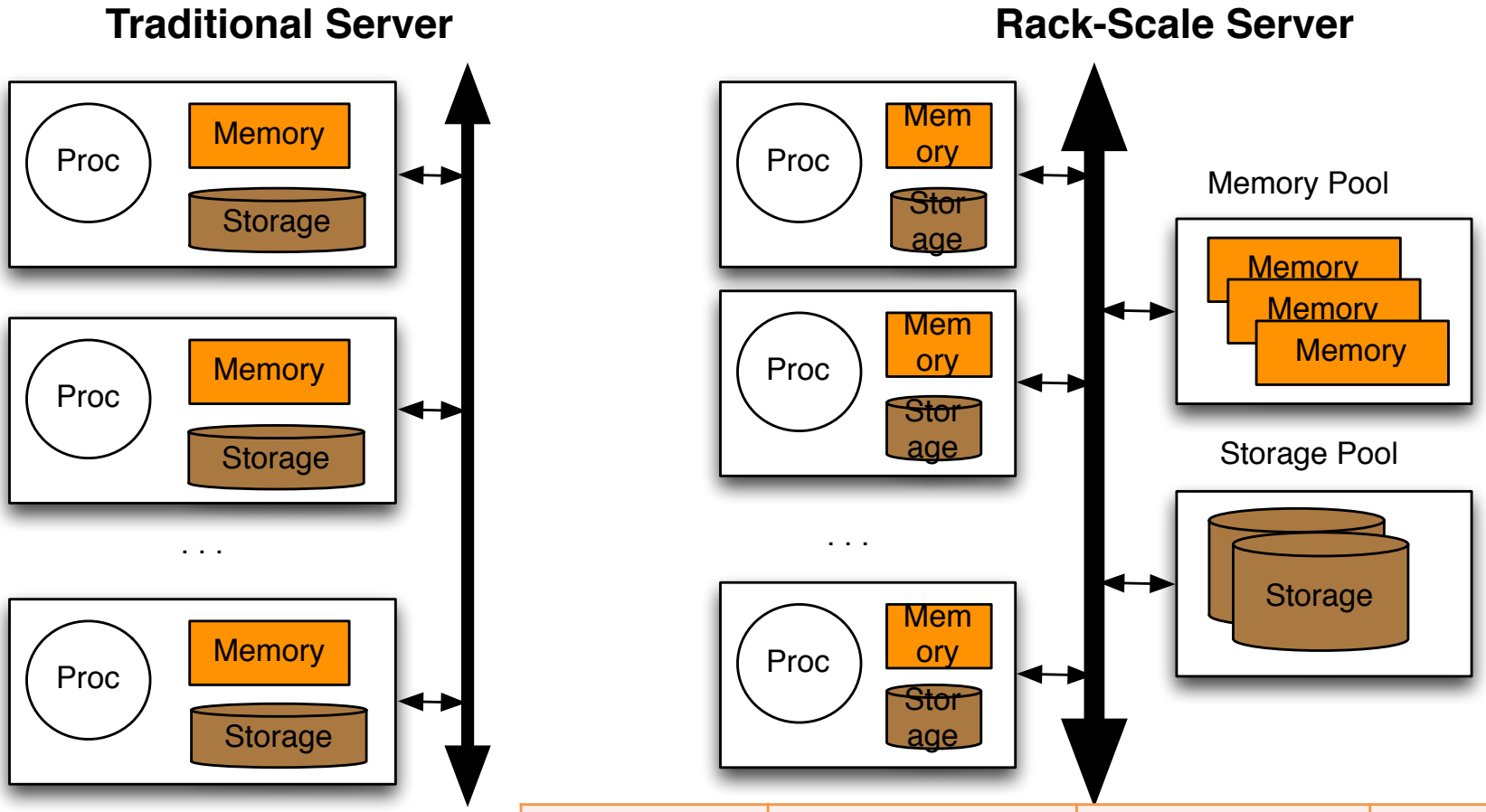
Problems with Intel SGX

- Use monolithic counters
  - Large memory overheads and slow
  - Why not use split counter?
- Use derivative of BMT: counter tree
- Single chip only
  - Won't work for whole memory, need DSM style
- Does not work with Persistent Memory
- Side channel ignored
  - Attackers will exploit this

Problems with Intel SGX

- Use monolithic counters
  - Large memory overheads and slow
  - Why not use split counter?
- Use derivative of BMT: counter tree
- Single chip only
  - Won't work for whole memory, need DSM style
- Does not work with Persistent Memory
- Side channel ignored
  - Attackers will exploit this

Traditional Server => Rack-Scale Server

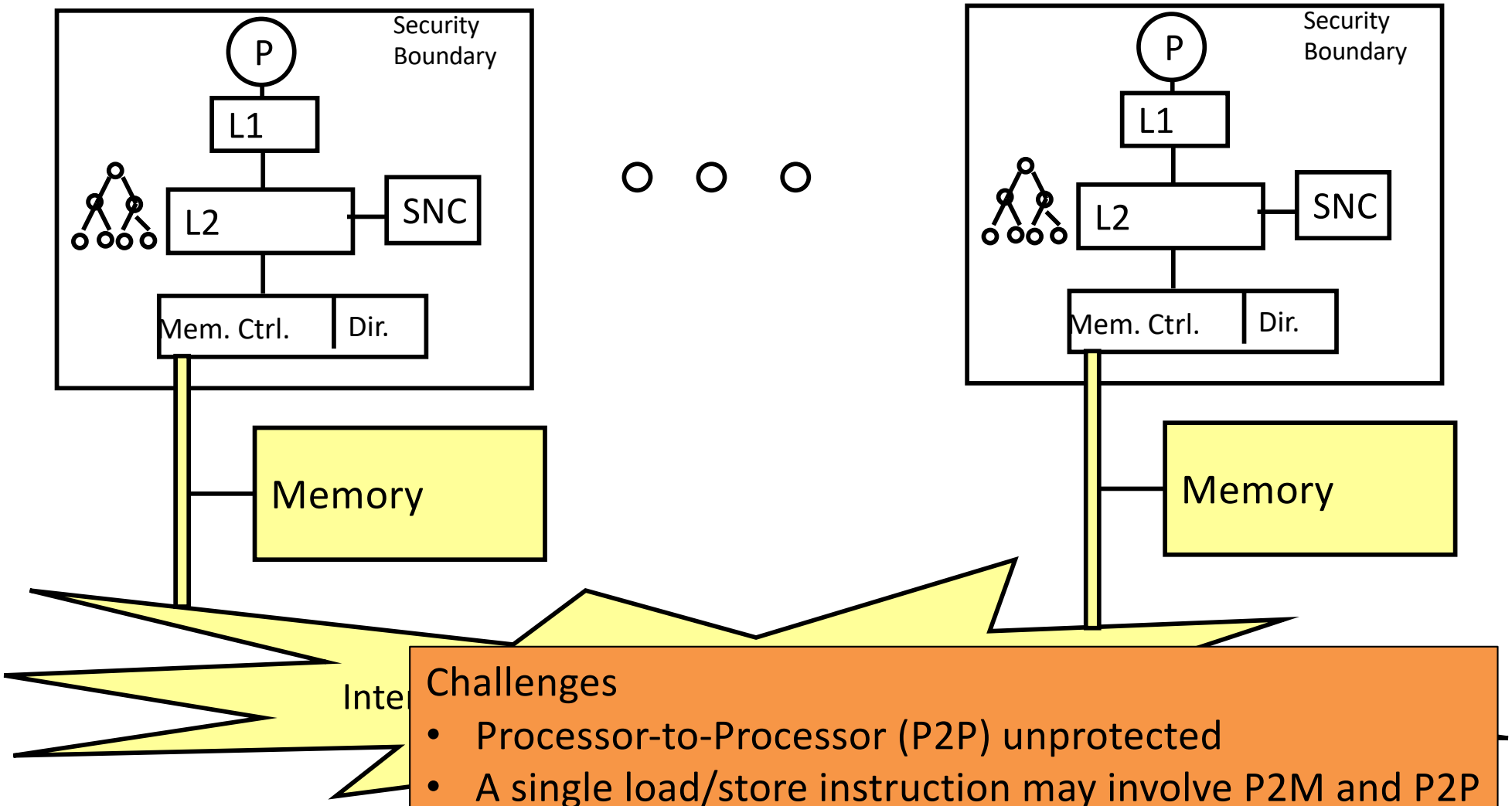


	Near (DRAM)	Near (NVM)	Far
Latency	1X	~5X	~30-50X
Bandwidth	1X	1X	0.1X
Capacity	1X	2X	10-100X

Key Milestones

- Rogers (2006): Distributed Shared Memory encryption

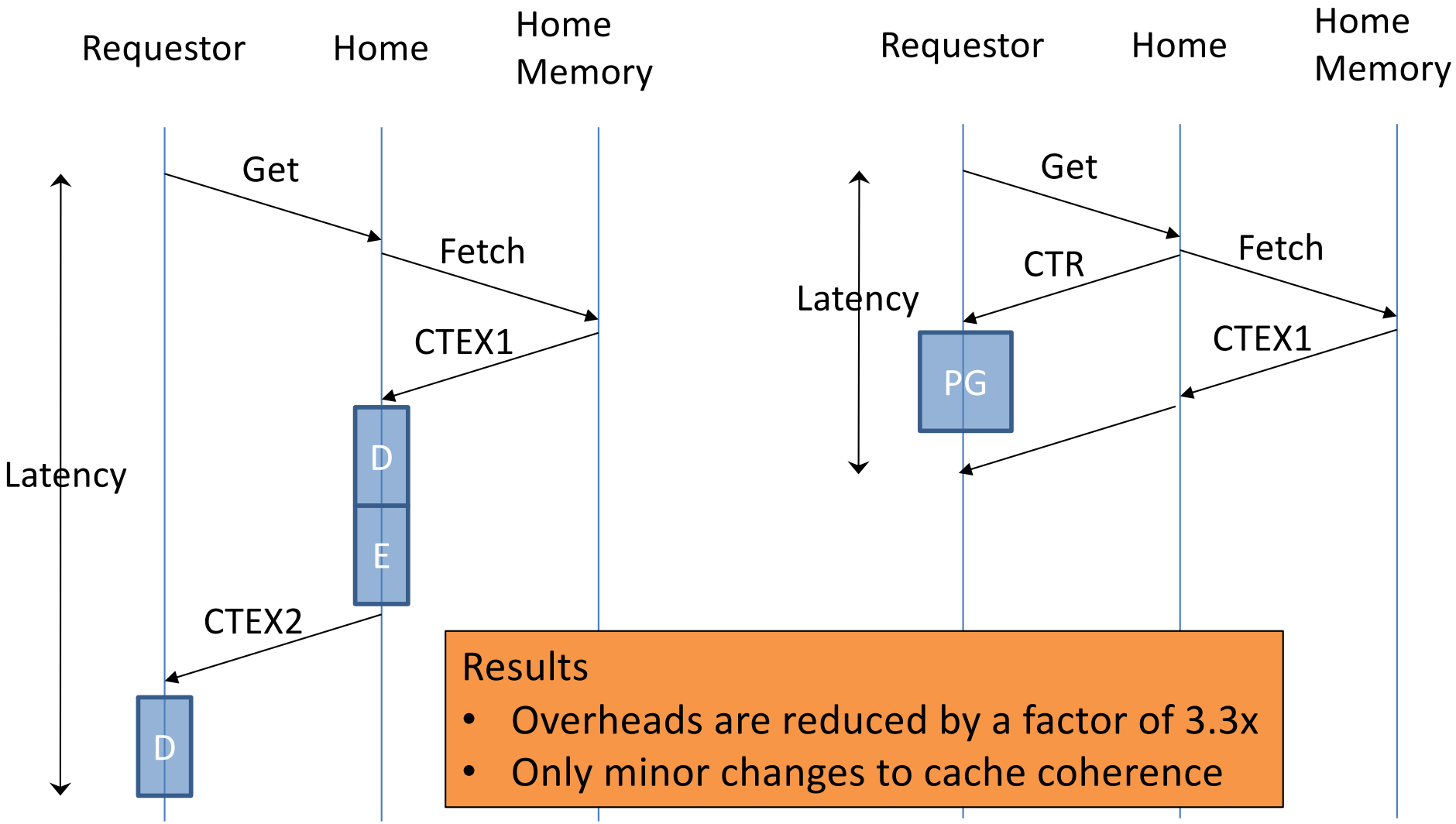
Architecture and Assumptions



Challenges

- Processor-to-Processor (P2P) unprotected
- A single load/store instruction may involve P2M and P2P
- Need a unified security protocol

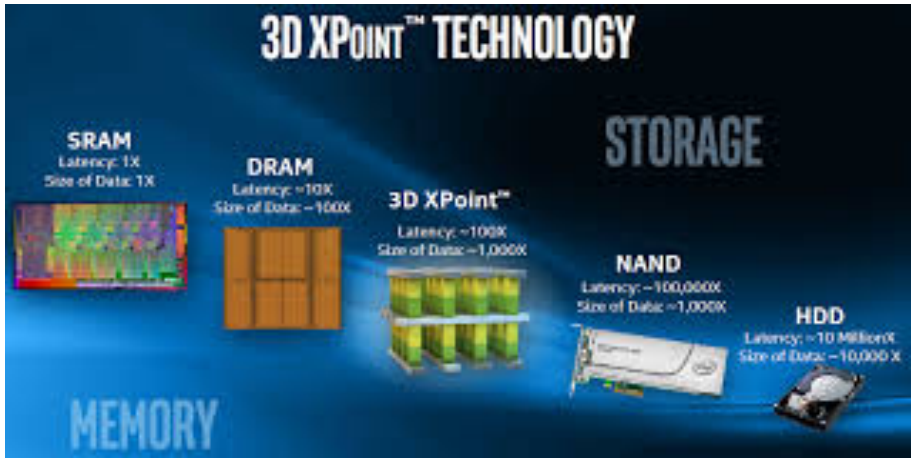
Huge Overheads from the Traditional Cache Coherence Protocol



Results

- Overheads are reduced by a factor of 3.3x
- Only minor changes to cache coherence

NVM and Storage Class Memory



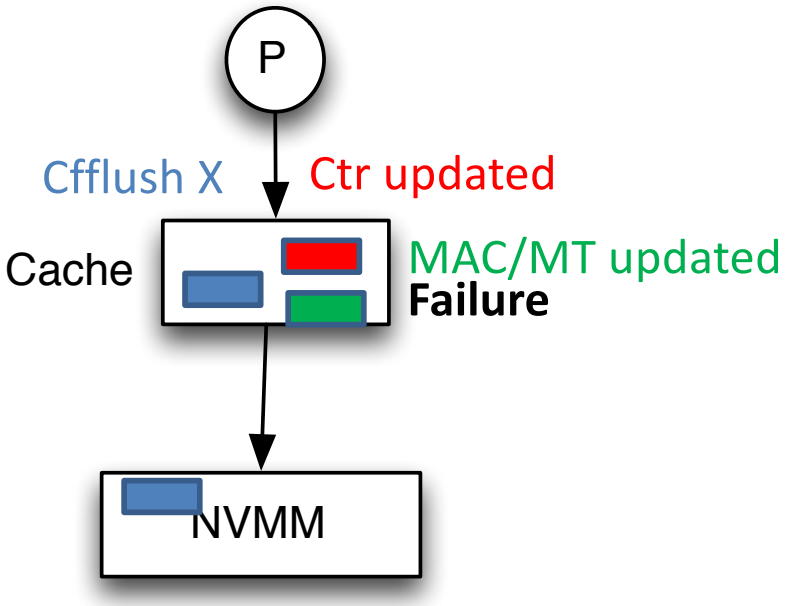
Intel 3D Xpoint (Optane):

- 20nm process
- SLC (1 bit/cell)
- 7 microsec latency
- 78,500 (70:30 random) read/write IOPS
- NVMe interface
- 375GB – 1.5TB

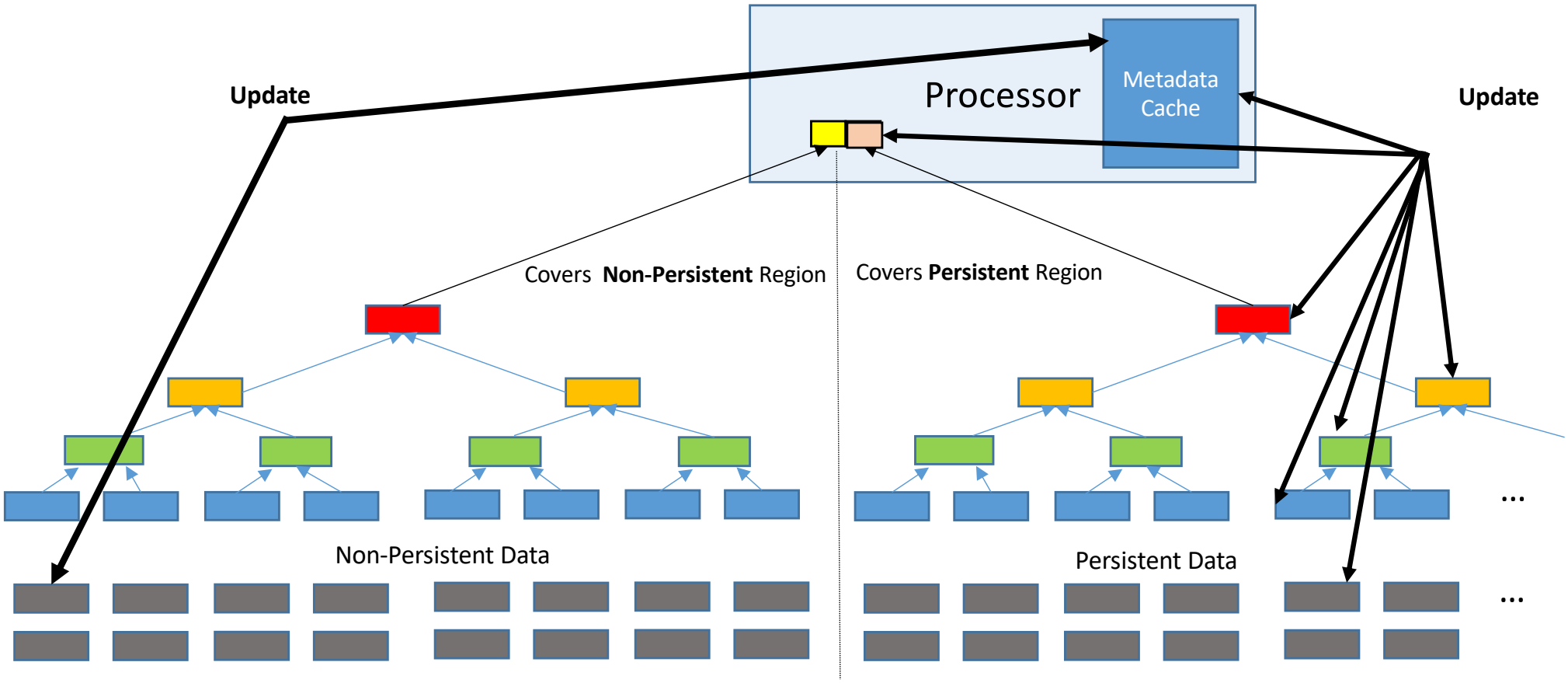
Source: Internet

Persistent Memory

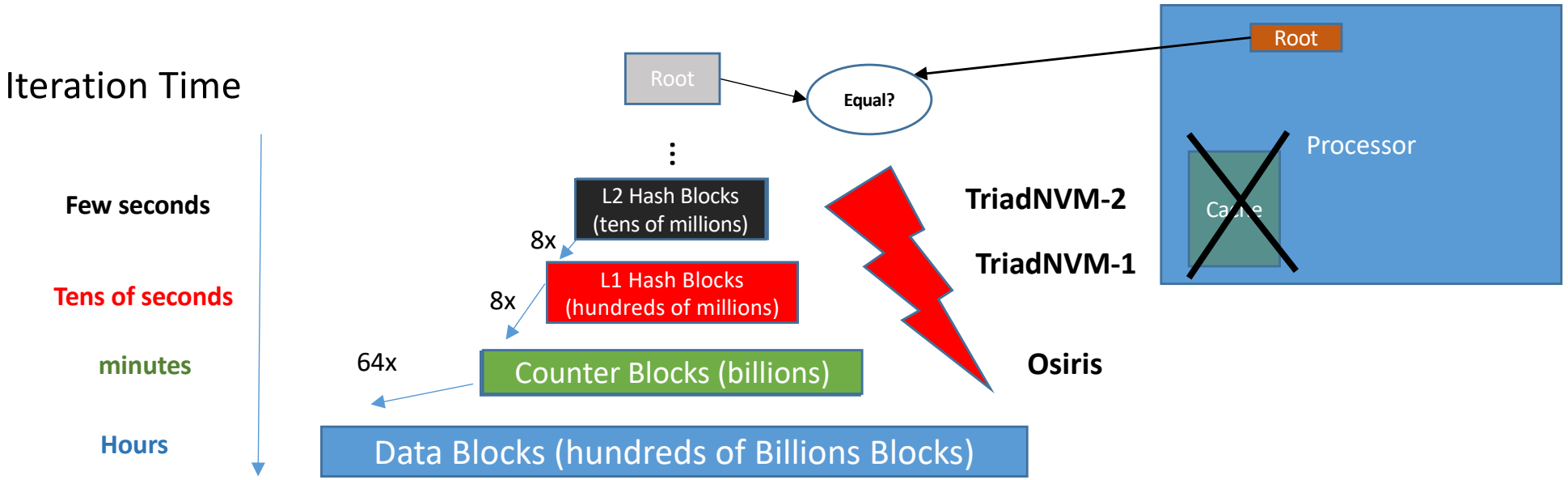
- Program relies on data to be recoverable after crash
- Counter and MT trees must be updated atomically w.r.t. to data
- Else data not recoverable upon crash
- Secure persistent memory [ISCA'19]



Optimizing Merkle Tree Persistence

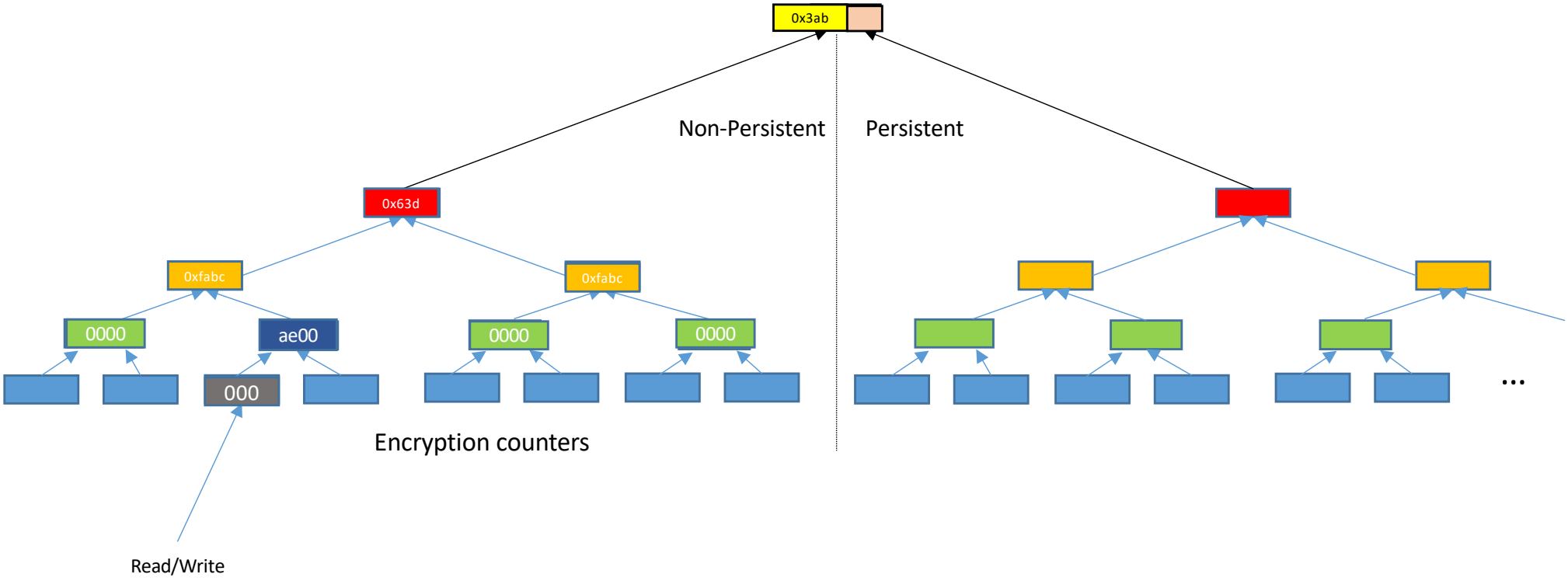


Recovery Time vs. Runtime Performance Overhead

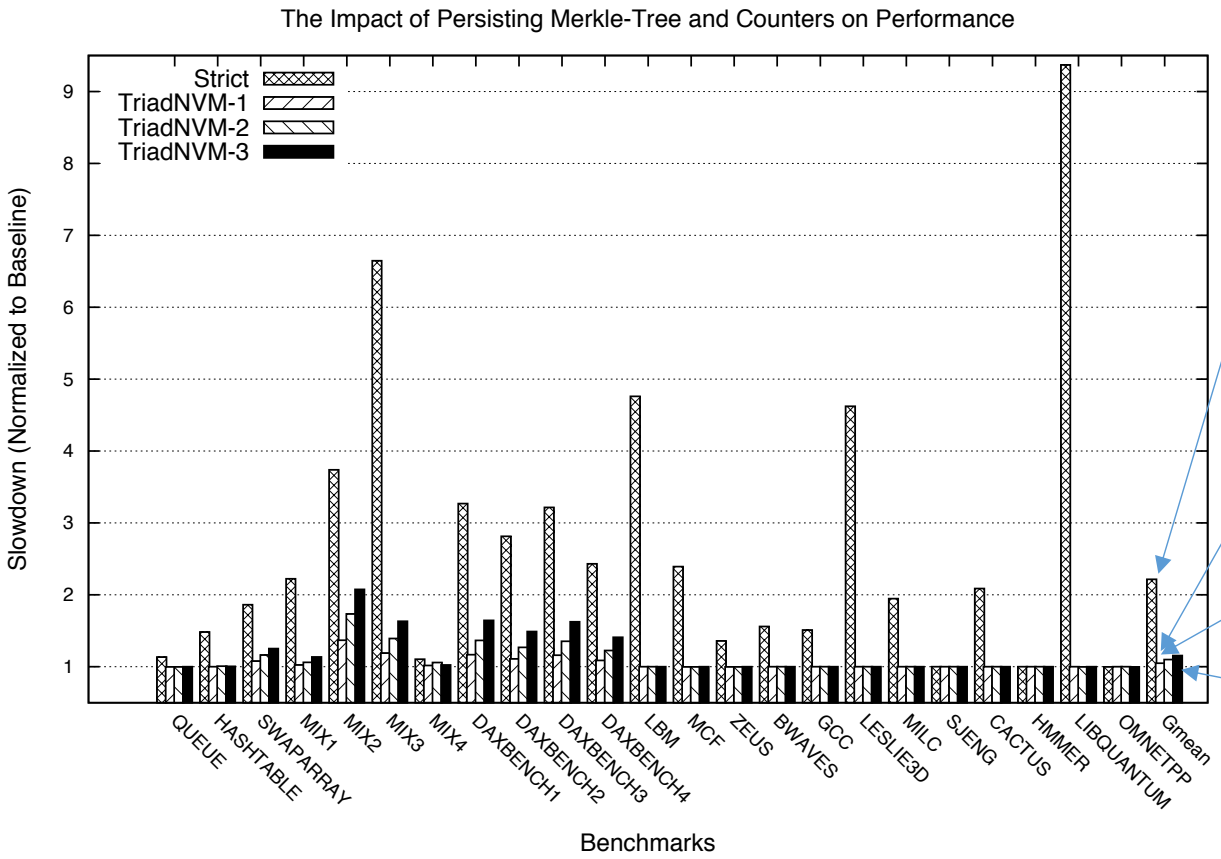


- Strictly Persisting upper levels reduces recovery time.
- In 8-ary tree and split-counter organization, only persisting the first two levels can speed up recovery time by 64*8= 512x times.
- Recovering 8TB memory can take more than 7 hours.

Reducing Initialization Time of Non-Persistent Region



Impact on Performance



Strict Persistence
121.5% performance overhead
Immediate recovery

TriadNVM-1
4.9% performance overhead
4 minutes to recover 8TB

TriadNVM-2
10% performance overhead
0.5 minute to recover 8TB

TriadNVM-3
15.6% performance overhead
3.83 seconds to recover 8TB

Side Channel Vulnerabilities

- "Side channel is the new buffer overflow"

  - private conversation with chief Scientific Advisor for National Security, UK

- Side channel vulnerability arises from implementation
- Current TEE has not addressed side channels
  - Physical: requires physical access to the system
    - Power
    - Differential fault
    - Electromagnetic (EM)
    - Memory access pattern
    - Etc.
  - Logical: does not require physical access
    - Cache
    - Data remanence

Page Fault Side Channel

- [Xu S&P'15]

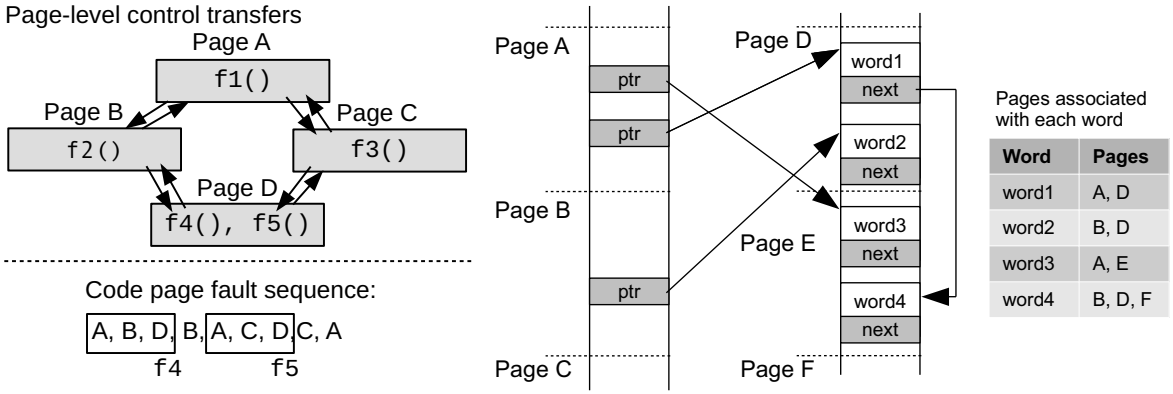
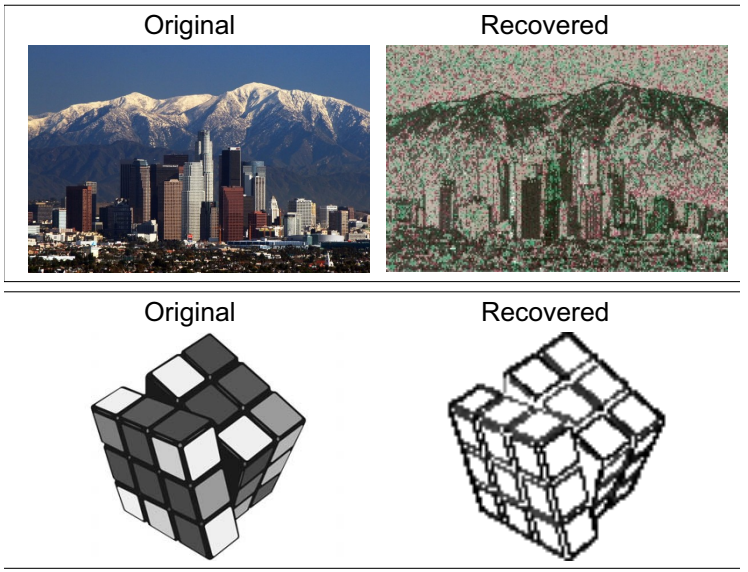


Fig. 4: The hash table in Hunspell.




```
Folklore, legends, myths and fairy tales
have followed childhood through the ages,
for every healthy youngster has a wholesome
and instinctive love for stories fantastic,
marvelous and manifestly unreal. The winged
fairies of Grimm and Andersen have brought
more happiness to childish hearts than all
other human creations.
```

```
folklore *legend* myths and fairy *tale*
have *follow* childhood through the *age*
for every healthy youngster has a wholesome
and instinctive love for [store] fantastic
marvelous and *manifest* unreal the [wine]
*fairy* of [grill] and Andersen have brought
more happiness to childish *heart* than all
other human *create*
```

Architecture-Related Side Channels

- Memory access pattern
- Cache side channel
  - Prime+Probe
  - Flush+Reload
  - Evict+Time
  - Flush+Flush
  - Prime+Abort
- Cache coherence side channel
- Branch predictor side channel

- No efficient protection, naïve approach is infeasible/impractical
- Affecting cloud servers down to IoT
- Full computing stack
  - Application, program, compiler, system, architecture, and hardware
  - Techniques must be composable

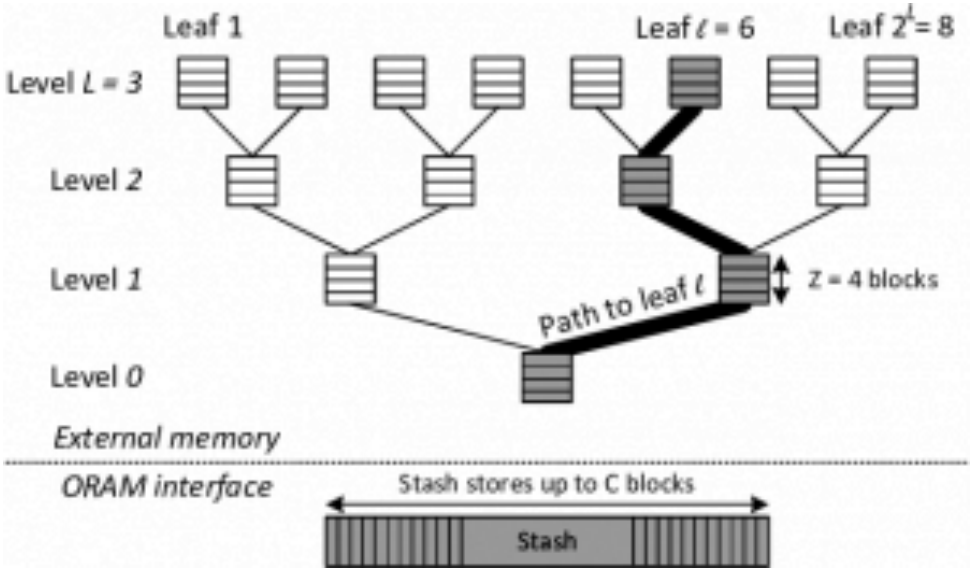
Memory Access Pattern

- Data leaks through access pattern, e.g.

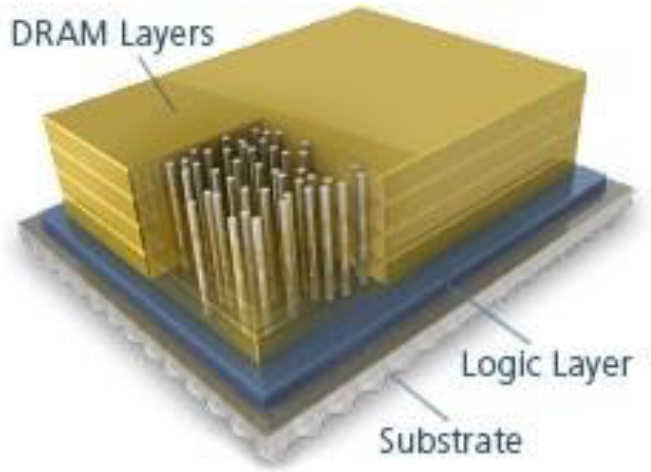
```
if (Data[i] == 0) {
  ... = A[i];
else
  ... = B[i];
}
```

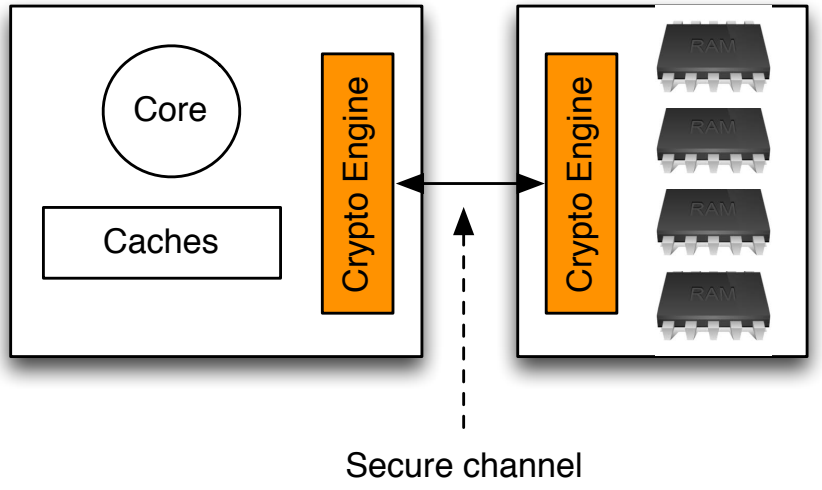
- By looking at access pattern we can reconstruct Data[i] values

- Current protection: ORAM
  - Block address randomized after each access



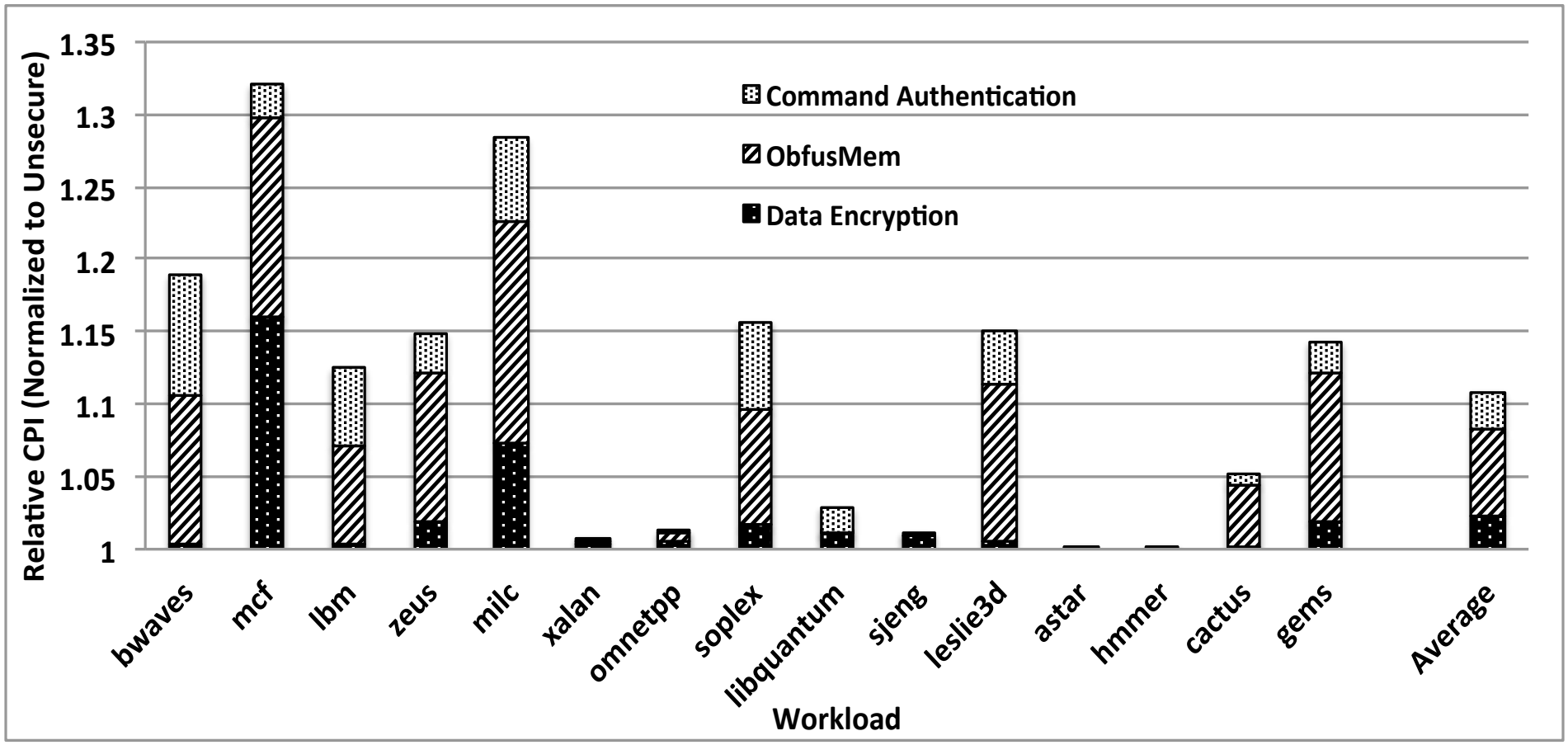
ObfusMem [ISCA '17]





- Premise
  - Memory becoming smart
    - 3D memory has logic layer
    - Memory interface packetized
    - We can put crypto engine in memory
- Secure channel between processor and memory
  - 10% overheads vs. 900% in ORAM

ObfusMem Performance Overheads



Conclusions

- SEE is needed more than ever
  - Hardware root of trust reduces attack surface
  - Shift from cloud to edge computing
  - NVM augmenting/replacing DRAM as main memory
- Industry effort (TEE) needs substantial improvements
  - Must consider multi-processors, side channels, persistent memory
- Great time to work on architecture support for security!

Thank you and I'd be happy to answer your questions

Yan.solihin@ucf.edu