# Transparent Memory Encryption and Authentication

**Mario Werner, *Thomas Unterluggauer*, Robert Schilling, David Schaffenrath, and Stefan Mangard,**
**IAIK, Graz University of Technology**

6. September 2017

# Content

- FPGA security: attackers with physical access
- IP protection: secure boot, FPGA bitfile encryption

# Content

- FPGA security: attackers with physical access
- IP protection: secure boot, FPGA bitfile encryption
- Missing protection for runtime data in RAM

# Content

- FPGA security: attackers with physical access
- IP protection: secure boot, FPGA bitfile encryption
- Missing protection for runtime data in RAM
- Open-source framework for RAM encryption and authentication

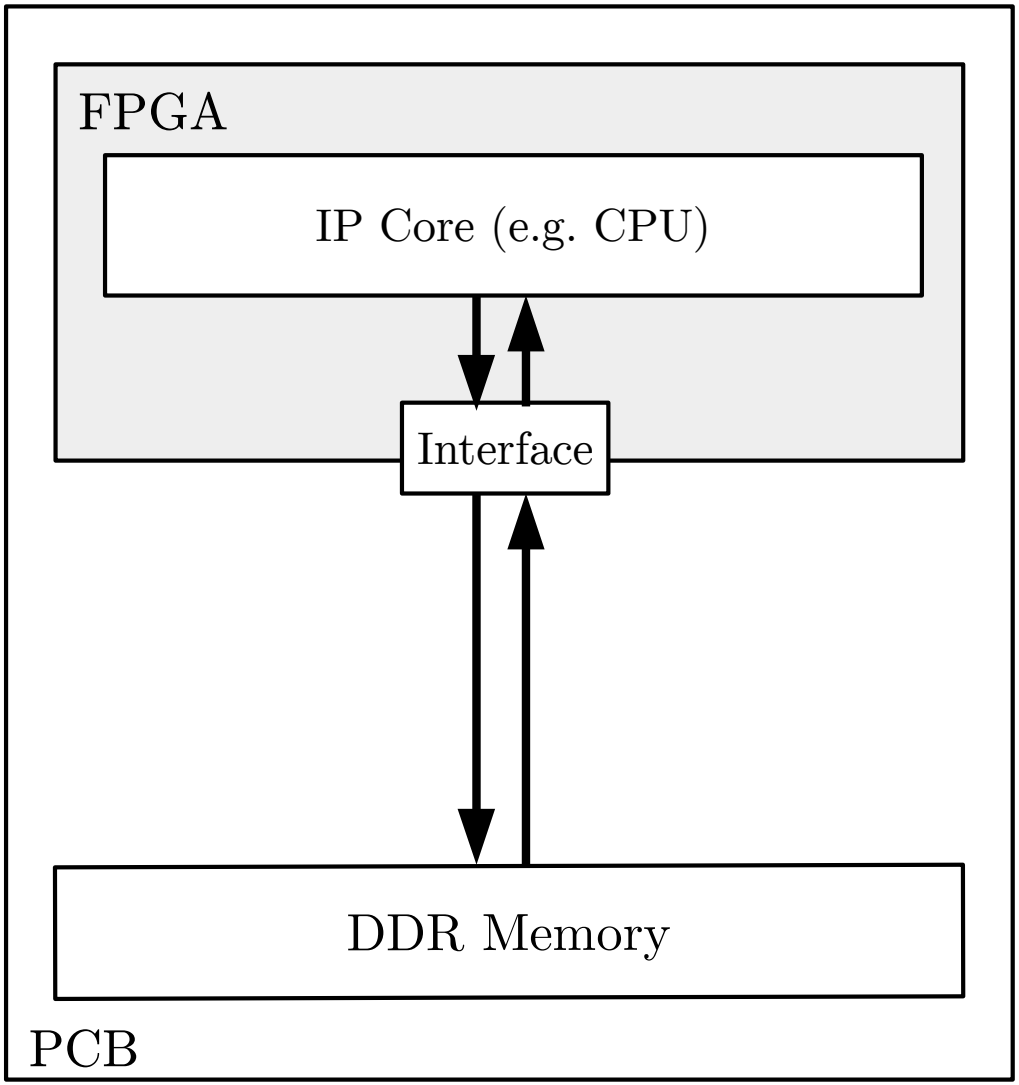
# Content

- FPGA security: attackers with physical access
- IP protection: secure boot, FPGA bitfile encryption
- Missing protection for runtime data in RAM
- Open-source framework for RAM encryption and authentication
  - Supports various cipher modes, e.g., AES-XTS
  - Evaluation on Xilinx Zynq-7020 SoC @ 50 MHz

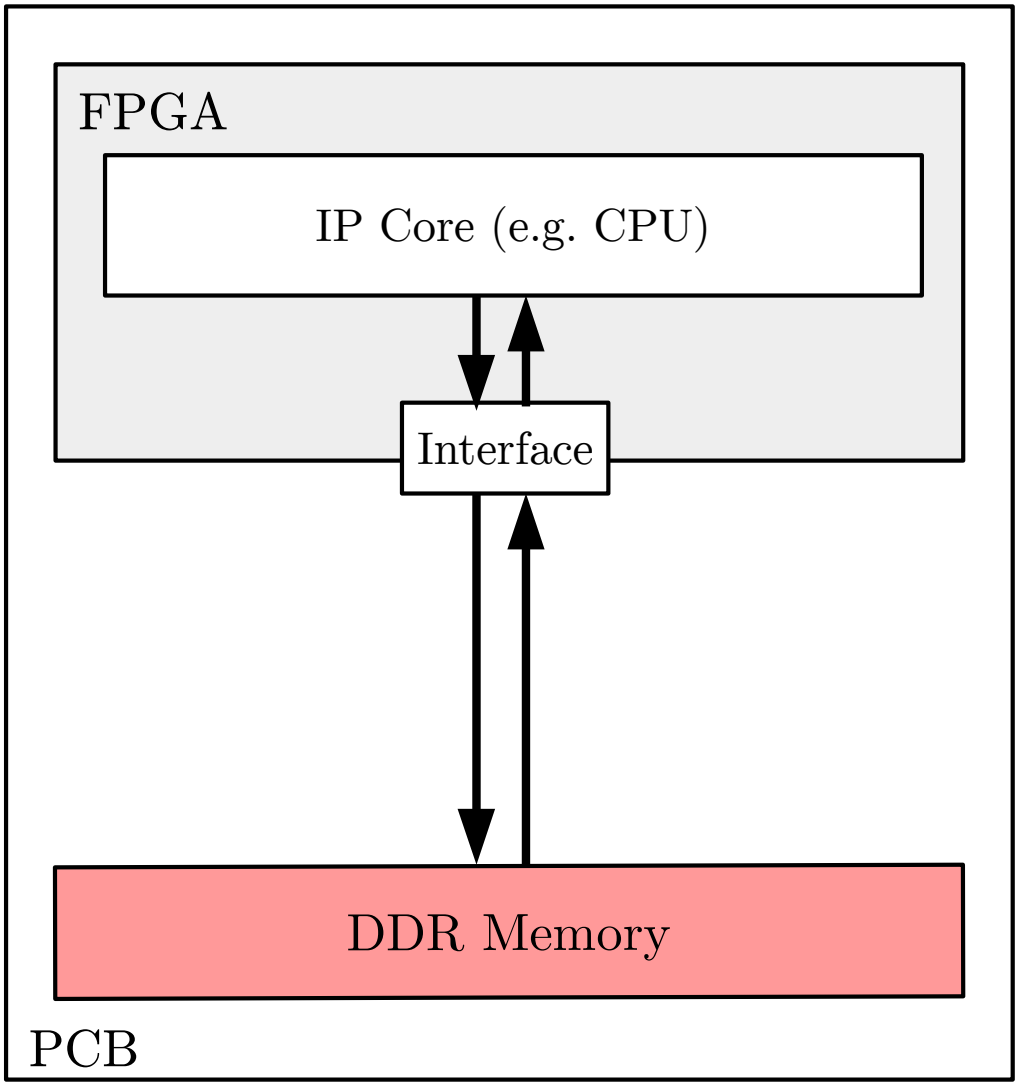
# Content

- FPGA security: attackers with physical access
- IP protection: secure boot, FPGA bitfile encryption
- Missing protection for runtime data in RAM
- Open-source framework for RAM encryption and authentication
  - Supports various cipher modes, e.g., AES-XTS
  - Evaluation on Xilinx Zynq-7020 SoC @ 50 MHz
  - Efficient pipeline: up to 187 MB/s
    - FPGA bus interface limit: 200 MB/sec
  - Authenticated encryption ASCON: 105 MB/s

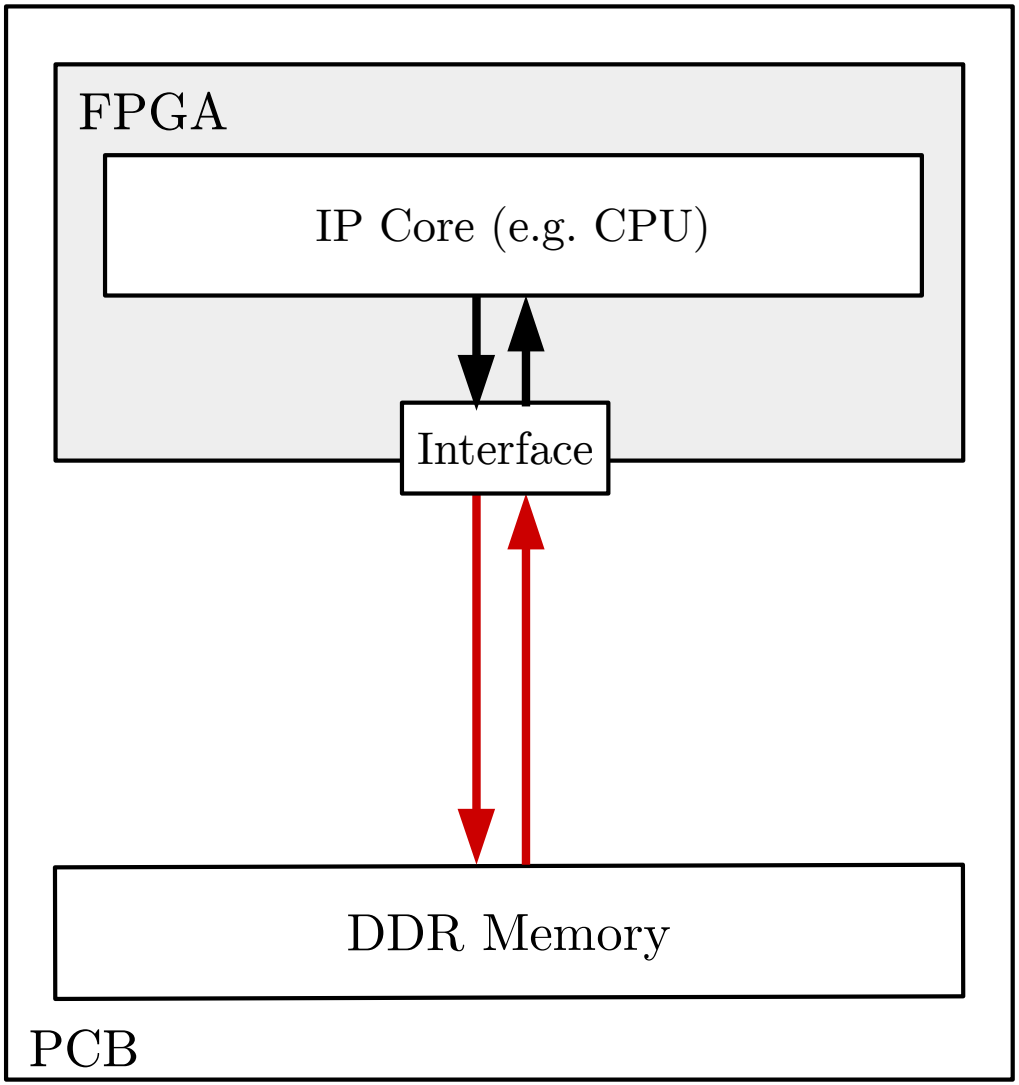
# Motivation



# Motivation



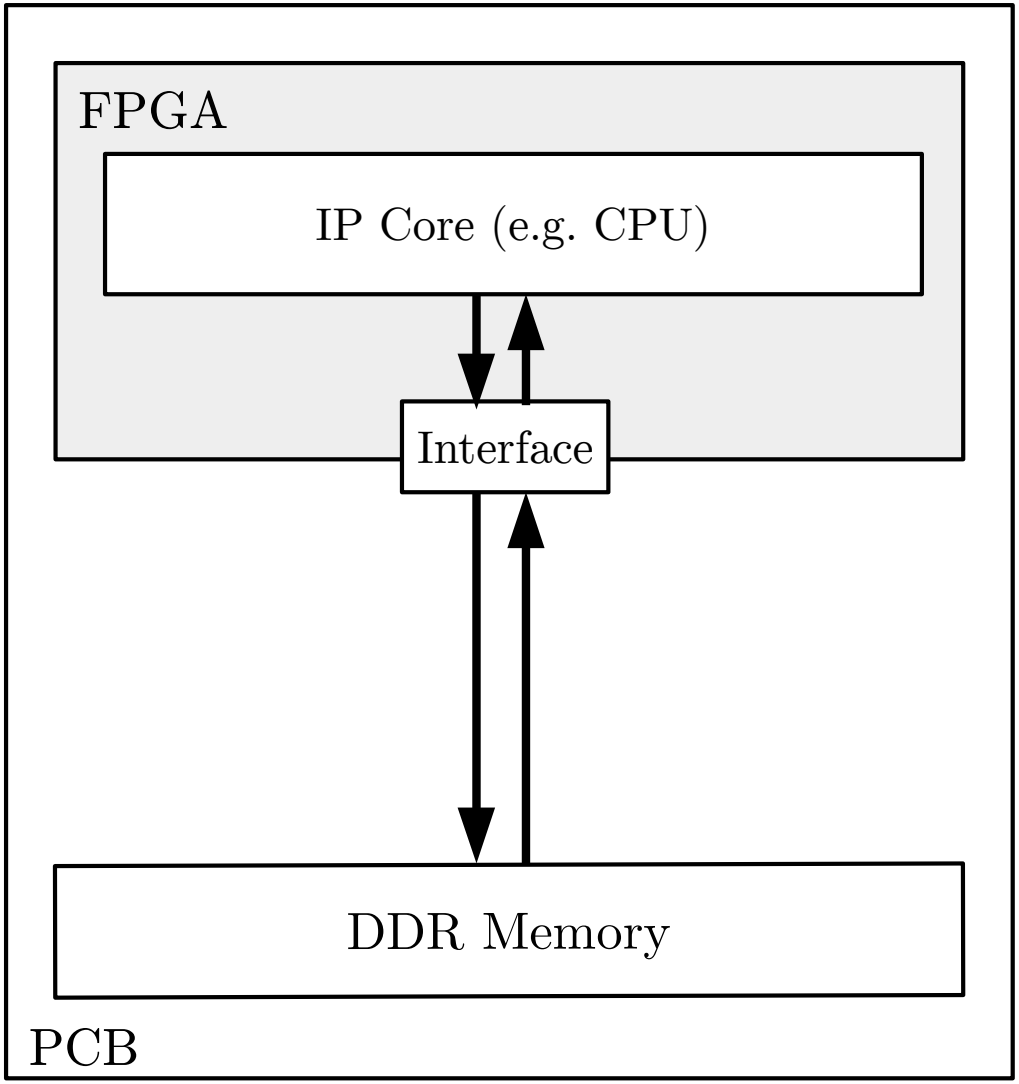
# Motivation



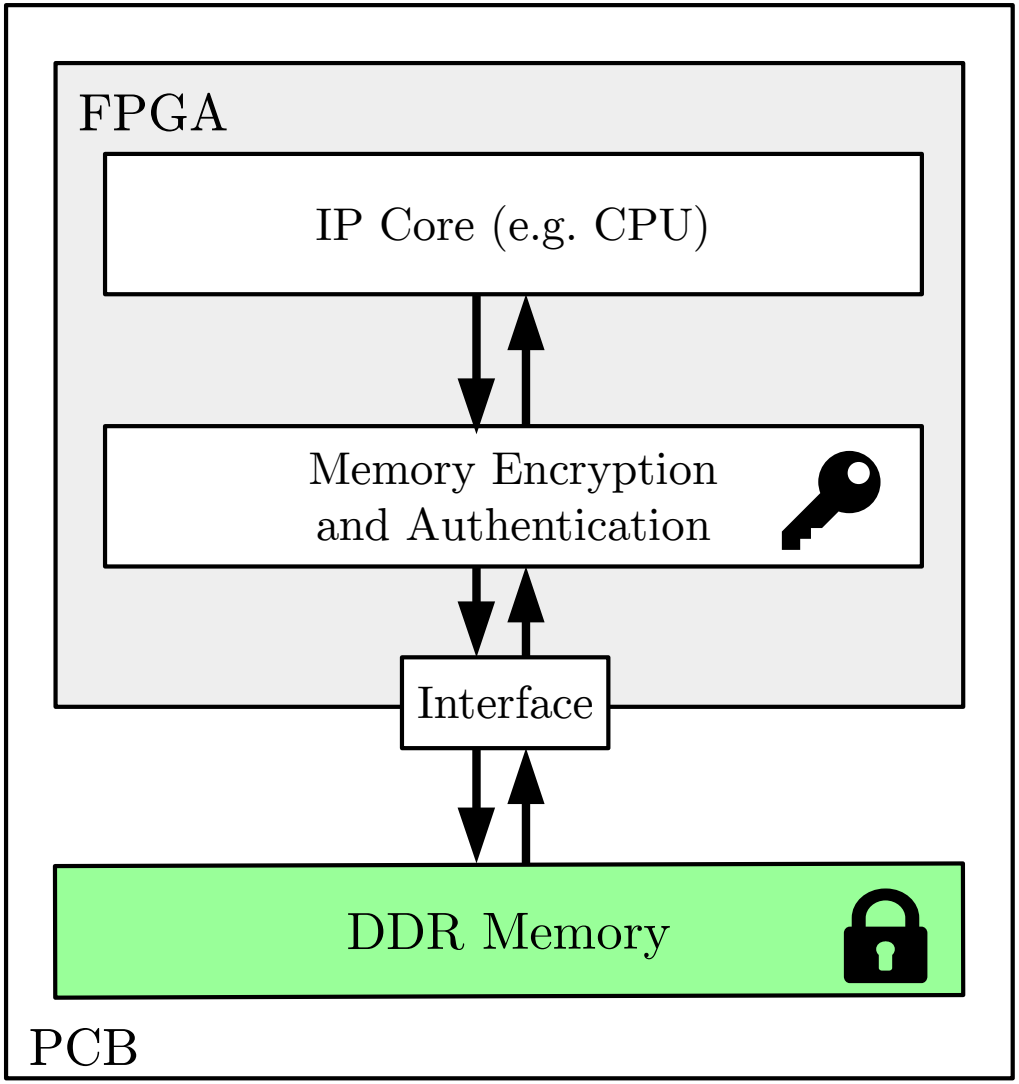
# Motivation

- RAM holds sensitive information and IP
- Hardly any protection of RAM during runtime
- **Need for encryption and authentication of RAM**

# Generic Concept

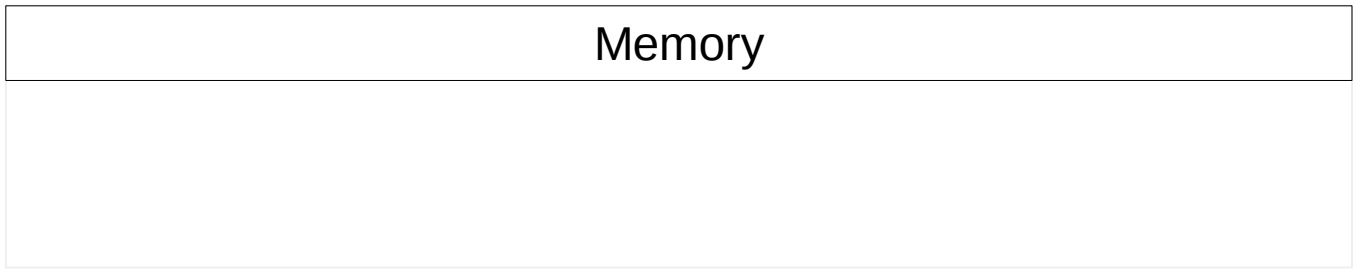


# Generic Concept



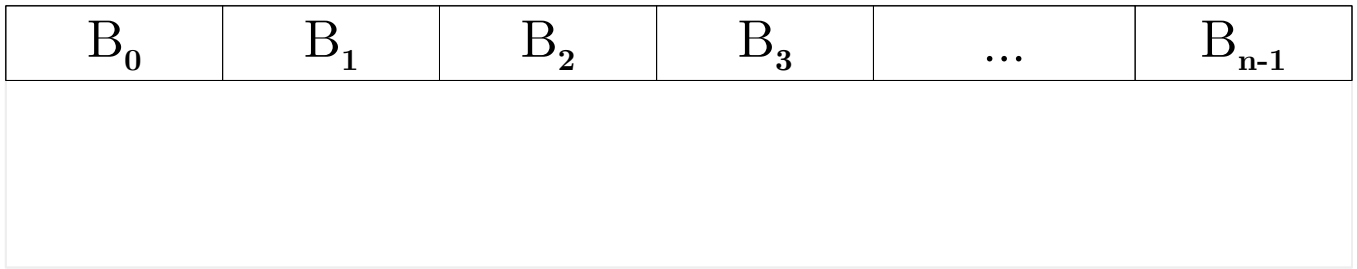
# Memory Encryption

- Split memory in blocks (e.g., sector or cache line size)



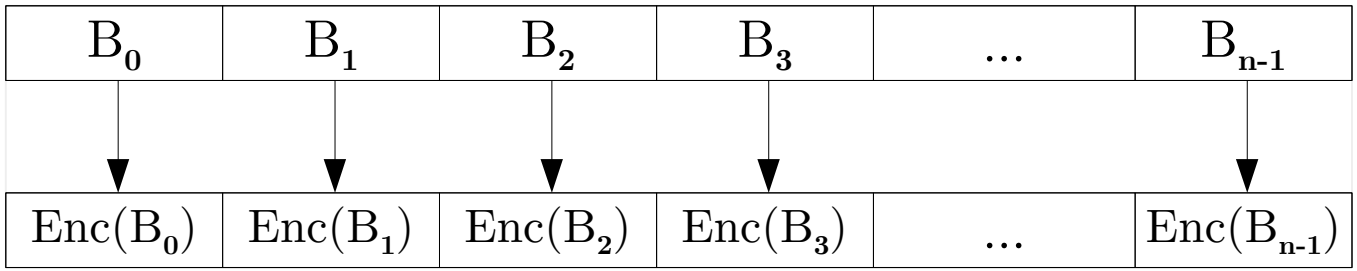
# Memory Encryption

- Split memory in blocks (e.g., sector or cache line size)



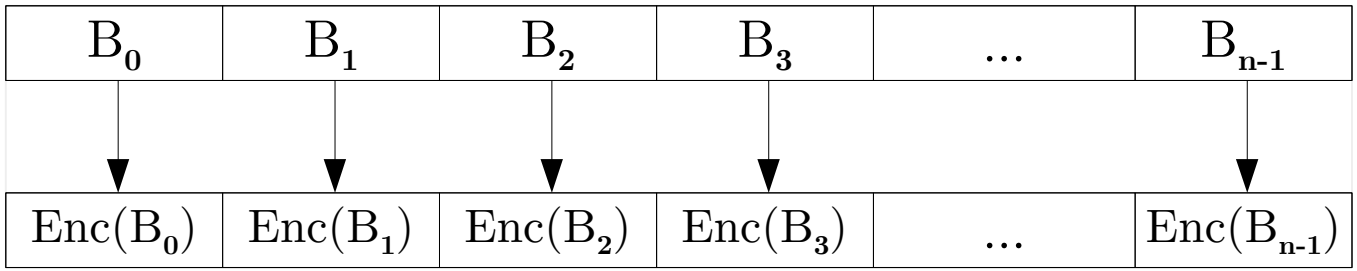
# Memory Encryption

- Split memory in blocks (e.g., sector or cache line size)
- Separate encryption of each memory block



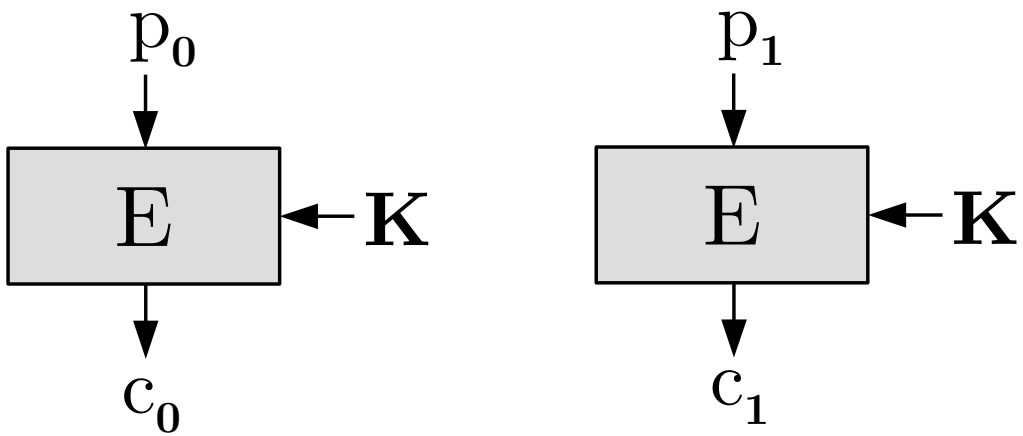
# Memory Encryption

- Split memory in blocks (e.g., sector or cache line size)
- Separate encryption of each memory block
  - ECB, CBC, XTS



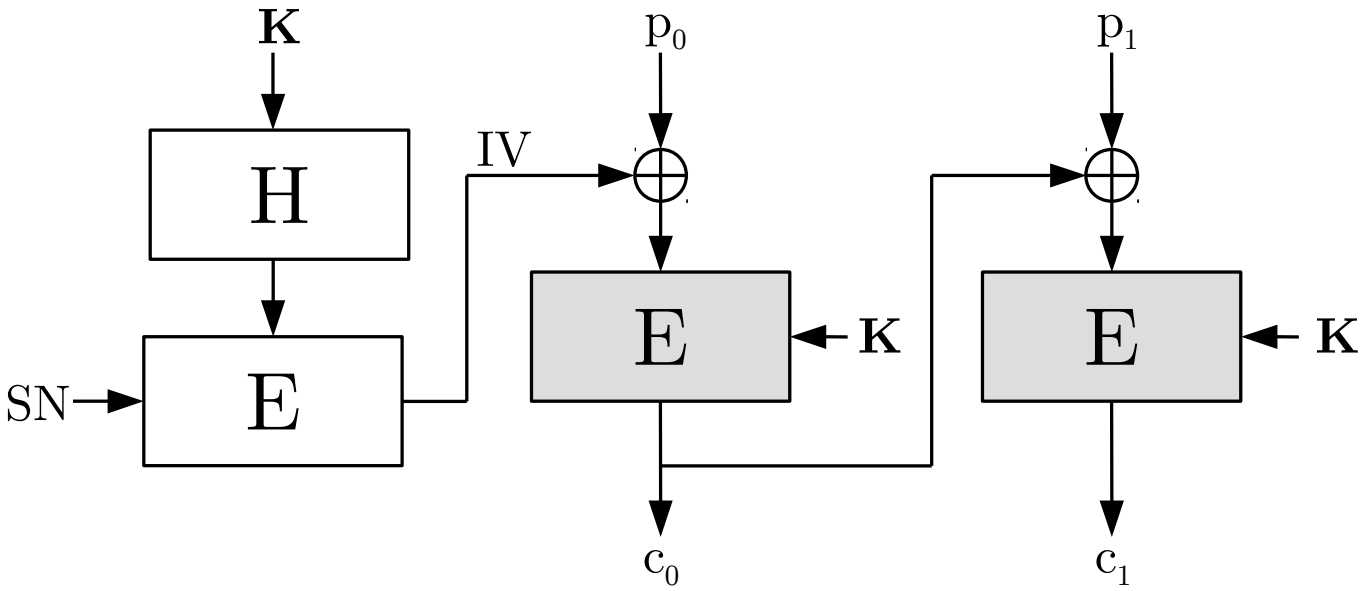
# Memory Encryption

- Split memory in blocks (e.g., sector or cache line size)
- Separate encryption of each memory block
  - **ECB**, CBC, XTS
  - Block $B$: $(p_0, p_1)$



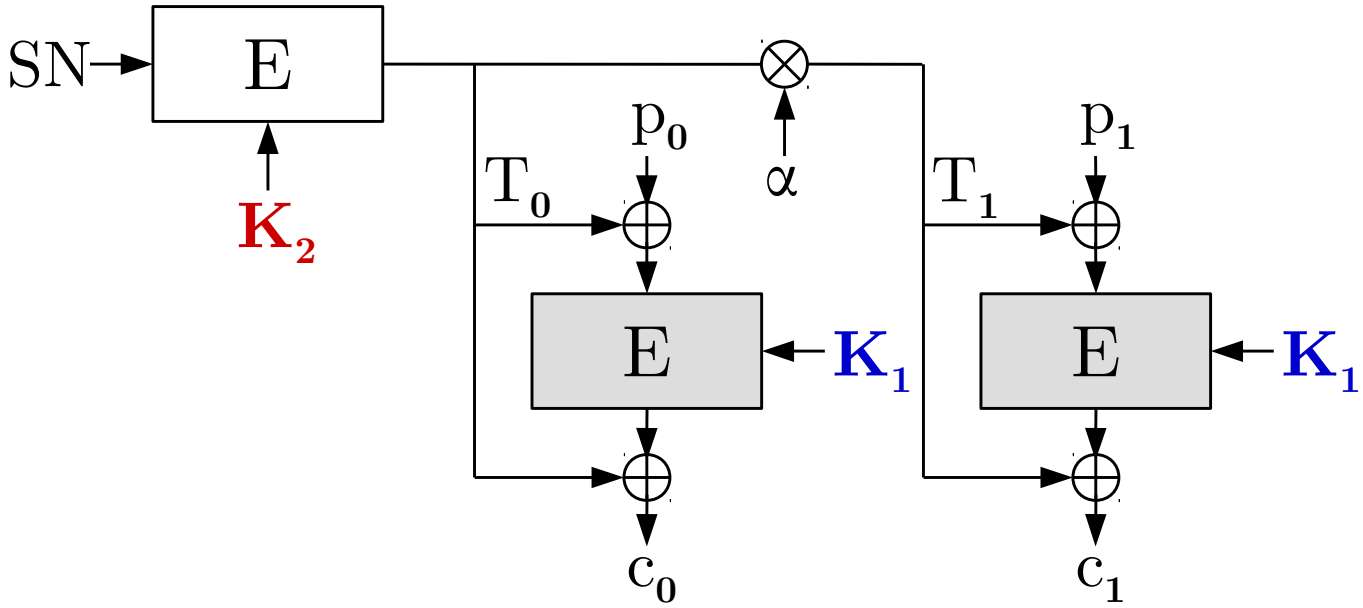
# Memory Encryption

- Split memory in blocks (e.g., sector or cache line size)
- Separate encryption of each memory block
  - ECB, **CBC**, XTS
  - Block $B$: $(p_0, p_1)$



# Memory Encryption

- Split memory in blocks (e.g., sector or cache line size)
- Separate encryption of each memory block
  - ECB, CBC, **XTS**
  - Block $B$: $(p_0, p_1)$



# Memory Encryption

- Split memory in blocks (e.g., sector or cache line size)
- Separate encryption of each memory block
  - ECB, CBC, XTS
  - Block ciphers: AES, PRINCE

# Memory Authentication

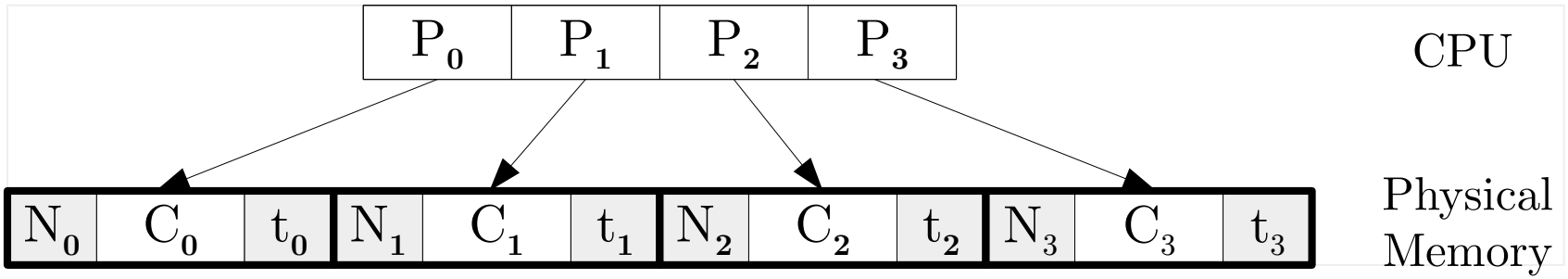
- Encryption does not protect against tampering
- Authenticated Encryption (AE) on memory blocks

# Memory Authentication

- Encryption does not protect against tampering
- Authenticated Encryption (AE) on memory blocks
  - Encryption: $(C, t) = Enc_K(P, N)$

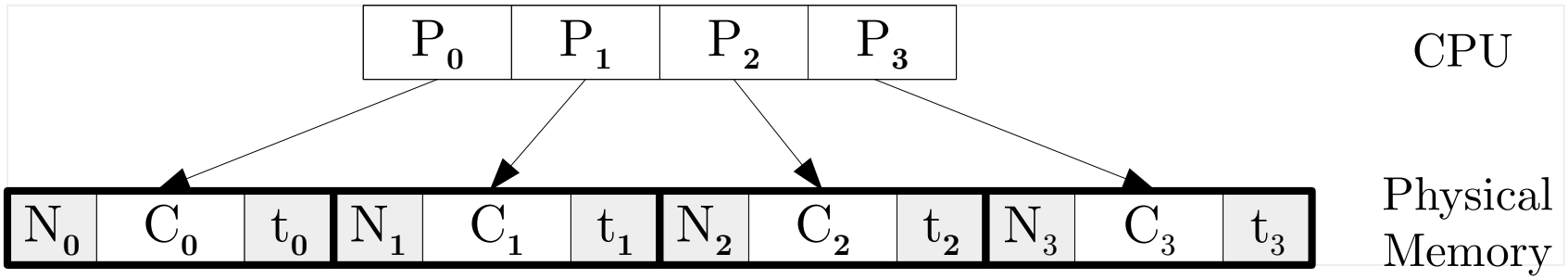
# Memory Authentication

- Encryption does not protect against tampering
- Authenticated Encryption (AE) on memory blocks
  - Encryption: $(C, t) = Enc_K(P, N)$



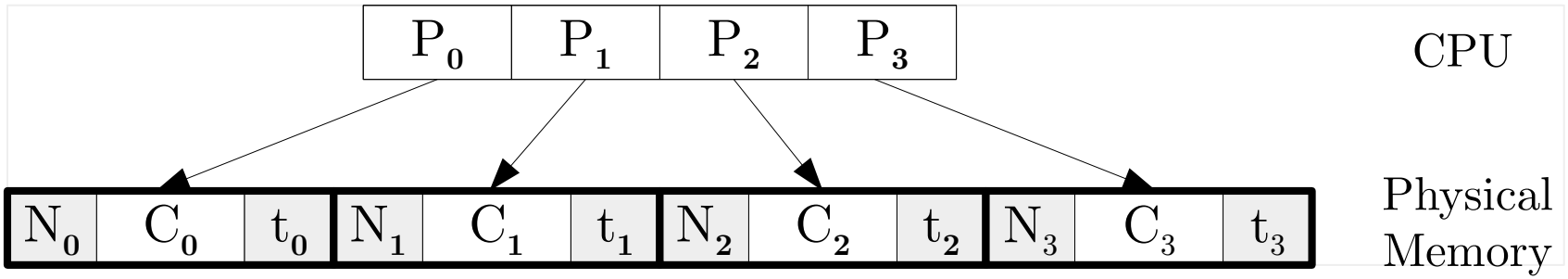
# Memory Authentication

- Encryption does not protect against tampering
- Authenticated Encryption (AE) on memory blocks
  - Encryption: $(C, t) = Enc_K(P, N)$
  - Decryption: $Dec_K(C, N, t) \in (P, \perp)$
  - ASCON



# Memory Authentication

- Encryption does not protect against tampering
- Authenticated Encryption (AE) on memory blocks
  - Encryption: $(C, t) = Enc_K(P, N)$
  - Decryption: $Dec_K(C, N, t) \in (P, \perp)$
  - ASCON
- Attacker cannot create valid $(C, N, t)$ without $K$

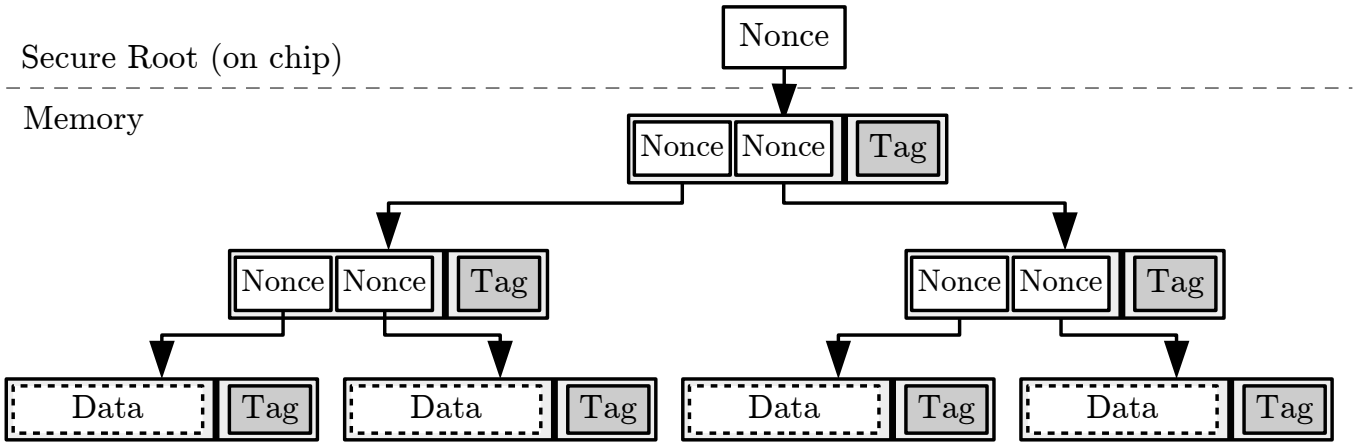


# Authentication Trees

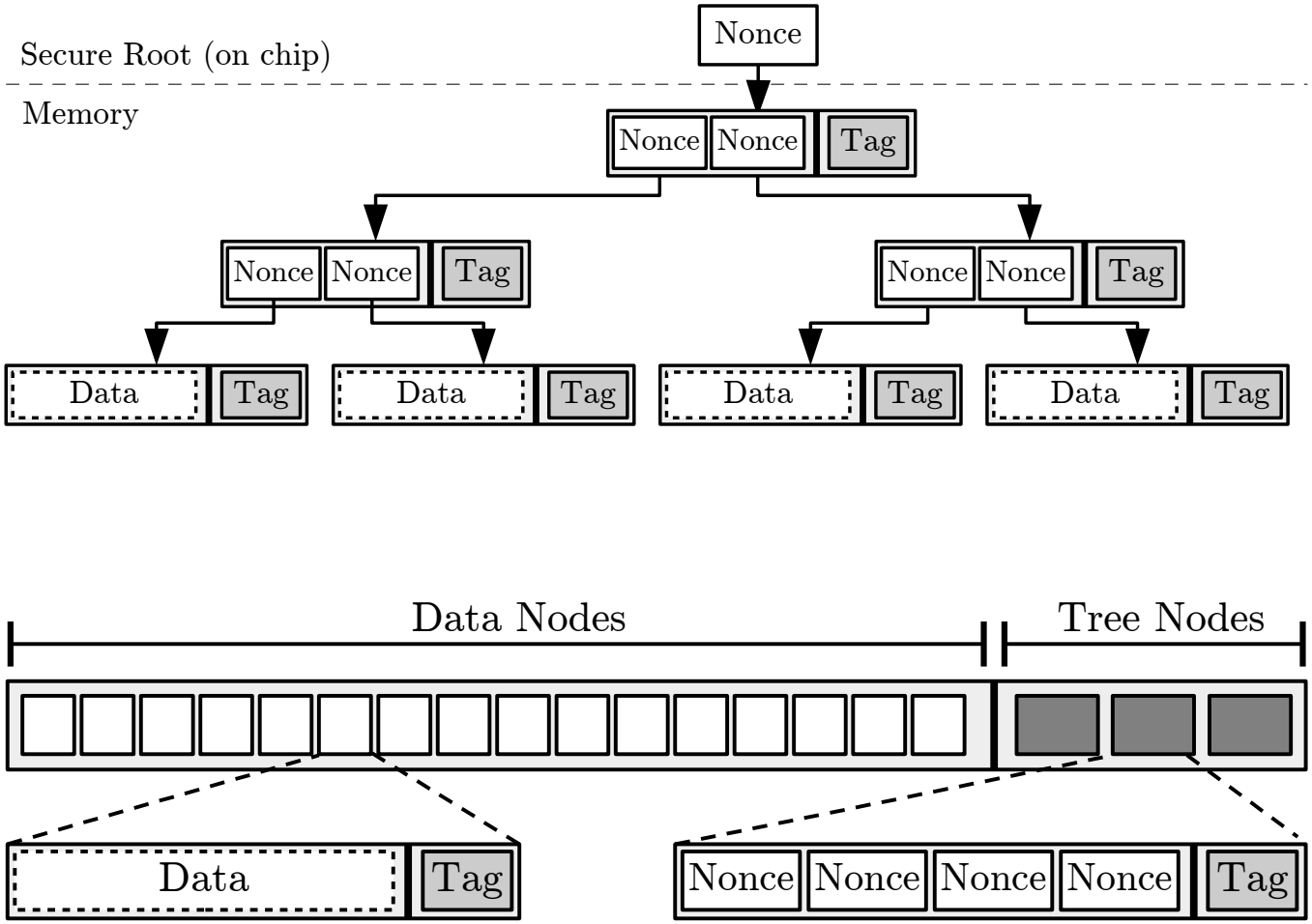
- Replay: attacker can restore old data
  - $(C, N, t)$ tuple remains valid forever
  - Prevented by authentication trees

# Authentication Trees

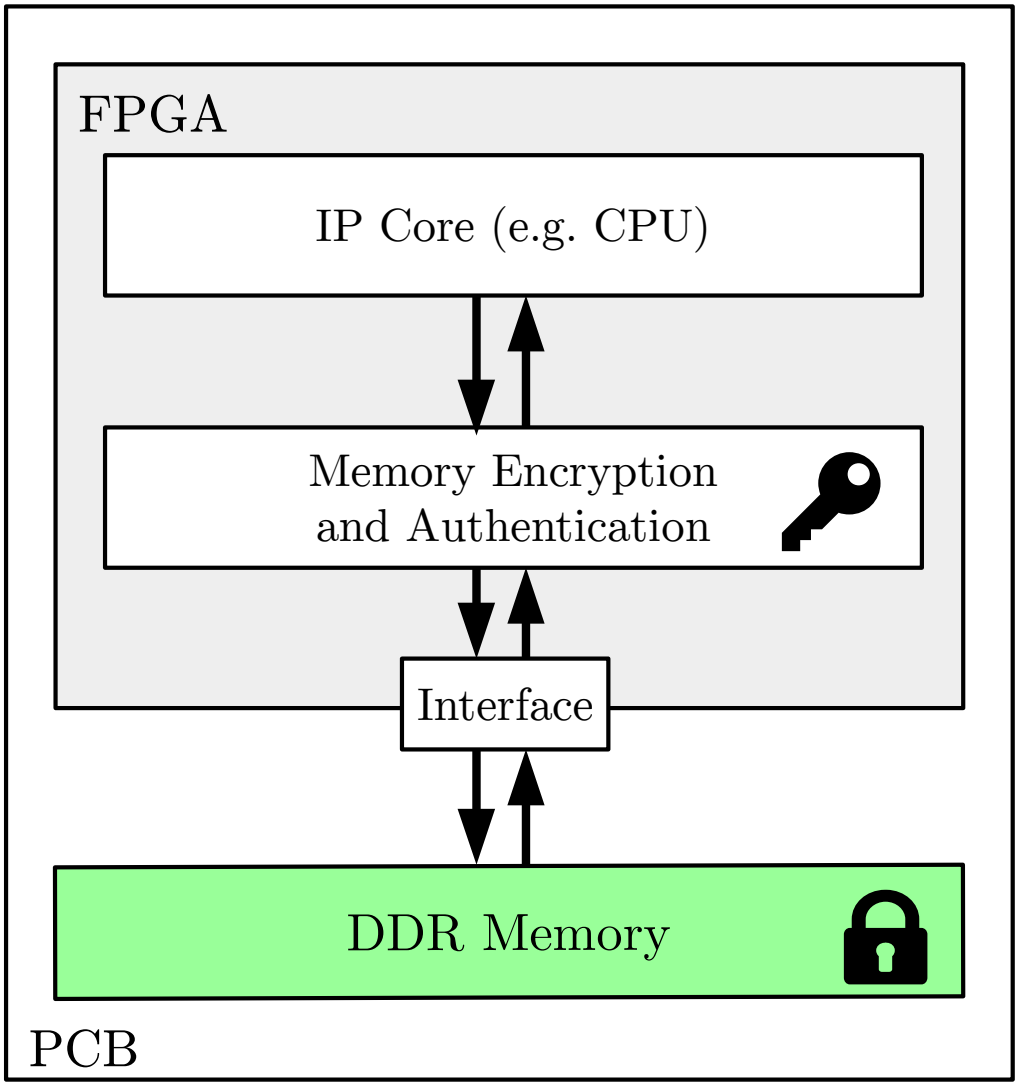
- Replay: attacker can restore old data
  - $(C, N, t)$ tuple remains valid forever
  - Prevented by authentication trees
- TEC tree based on ASCON



# Authentication Trees



# Implementation



# Challenges

- Memory requests from the master without constraints

# Challenges

- Memory requests from the master without constraints
- Requirements specific to the cryptographic mode:
  - Alignment
  - Block size
  - Metadata (counters, nonces, tags) and layout

# Challenges

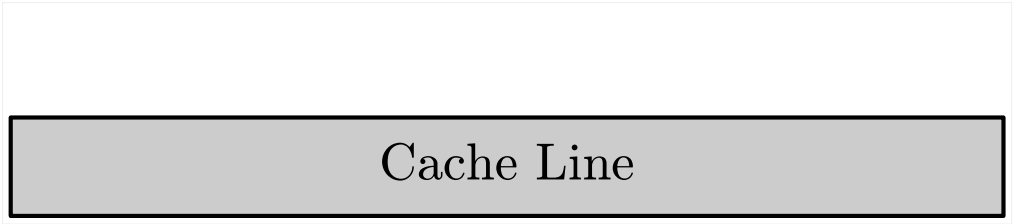
- Memory requests from the master without constraints
- Requirements specific to the cryptographic mode:
  - Alignment
  - Block size
  - Metadata (counters, nonces, tags) and layout
- Bus width mismatches
  - Master bus
  - Memory bus
  - Internal data stream
    - Aligned with cryptography

# Challenges (2)

- Specifics of AXI4 interfaces:
  - Write strobes
  - Narrow and wrapping bursts

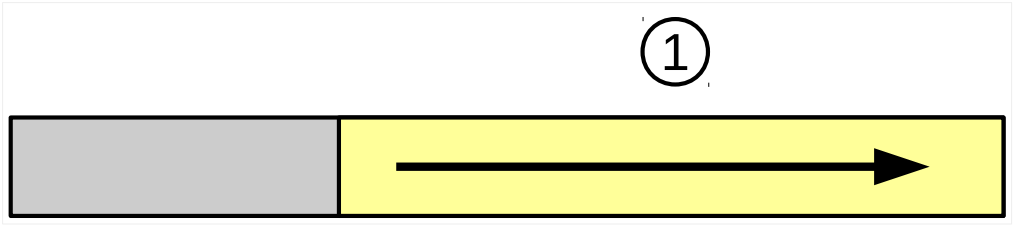
# Challenges (2)

- Specifics of AXI4 interfaces:
  - Write strobes
  - Narrow and wrapping bursts



# Challenges (2)

- Specifics of AXI4 interfaces:
  - Write strobes
  - Narrow and wrapping bursts



# Challenges (2)

- Specifics of AXI4 interfaces:
  - Write strobes
  - Narrow and wrapping bursts

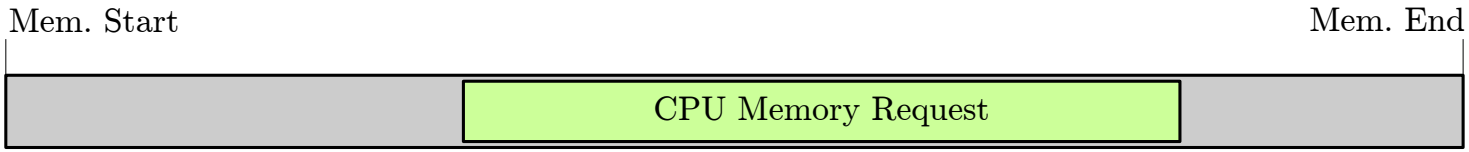
# Approach

- Framework consisting of modular building blocks
  - Bus interfaces
  - Request modification
  - En-/Decryption modules
  - Data stream modifications
  - Support (synchronization, rate conversions)

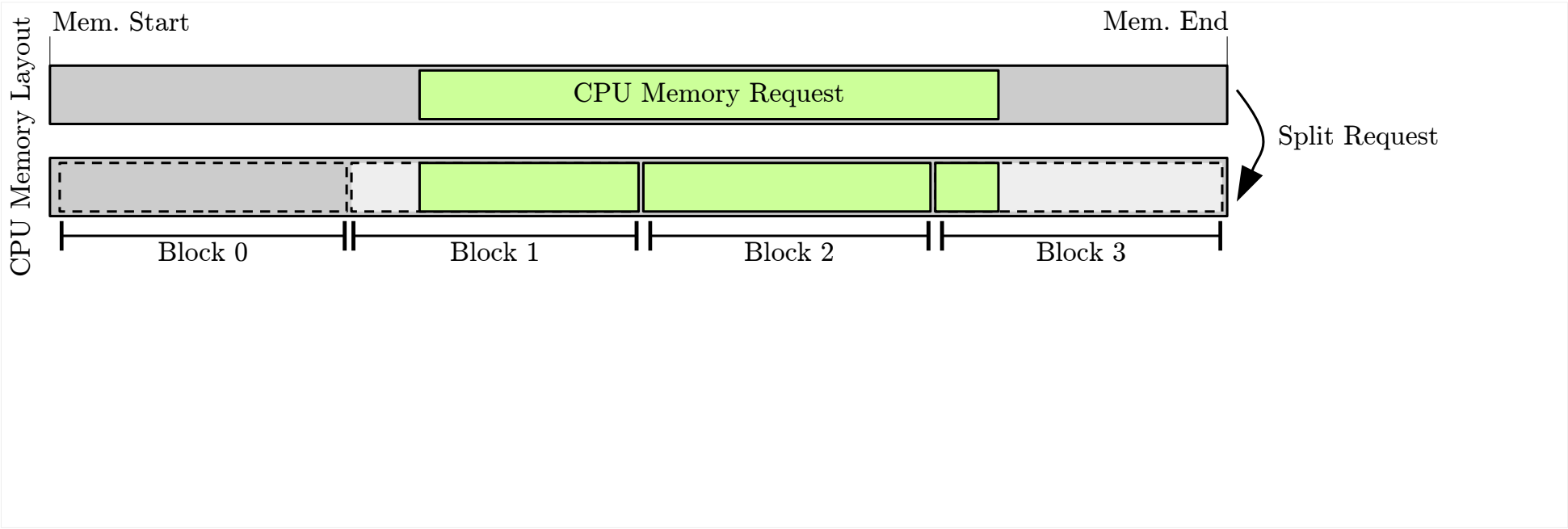
# Approach

- Framework consisting of modular building blocks
  - Bus interfaces
  - Request modification
  - En-/Decryption modules
  - Data stream modifications
  - Support (synchronization, rate conversions)
- Fully synchronized unidirectional interconnection

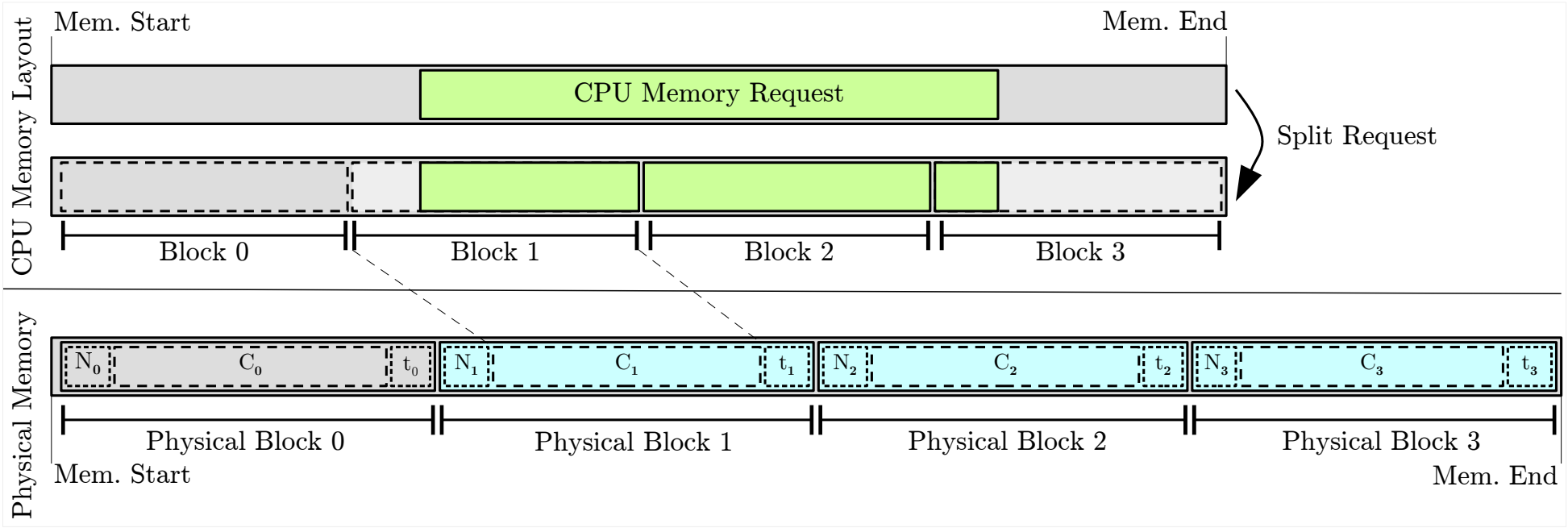
# Request Modification



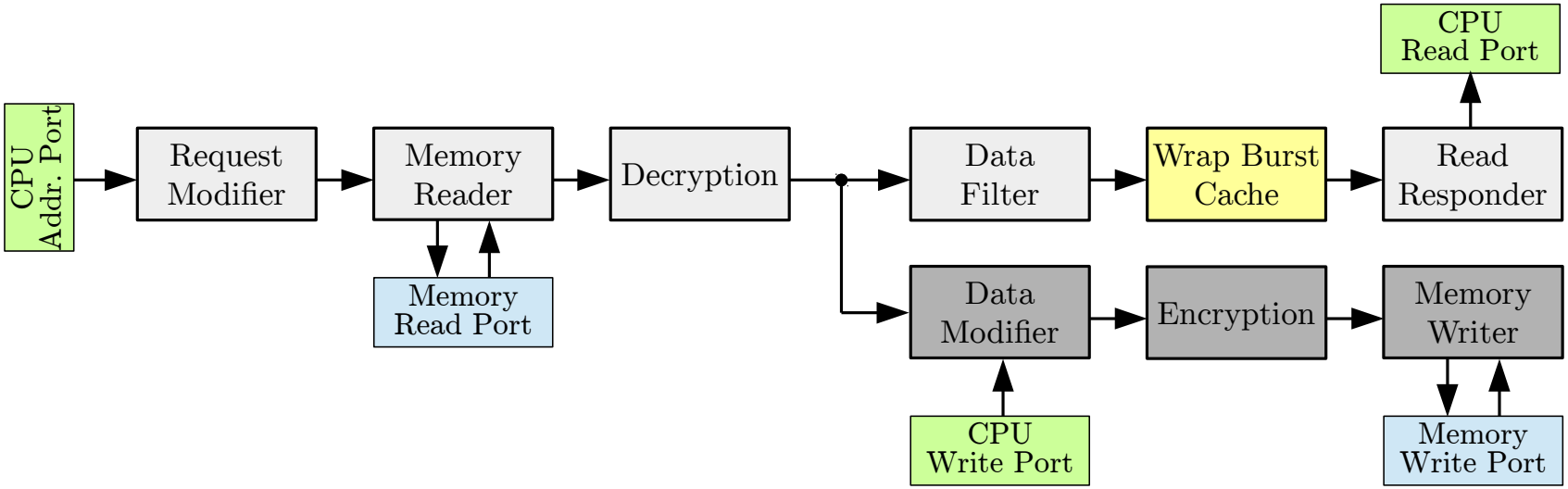
# Request Modification



# Request Modification

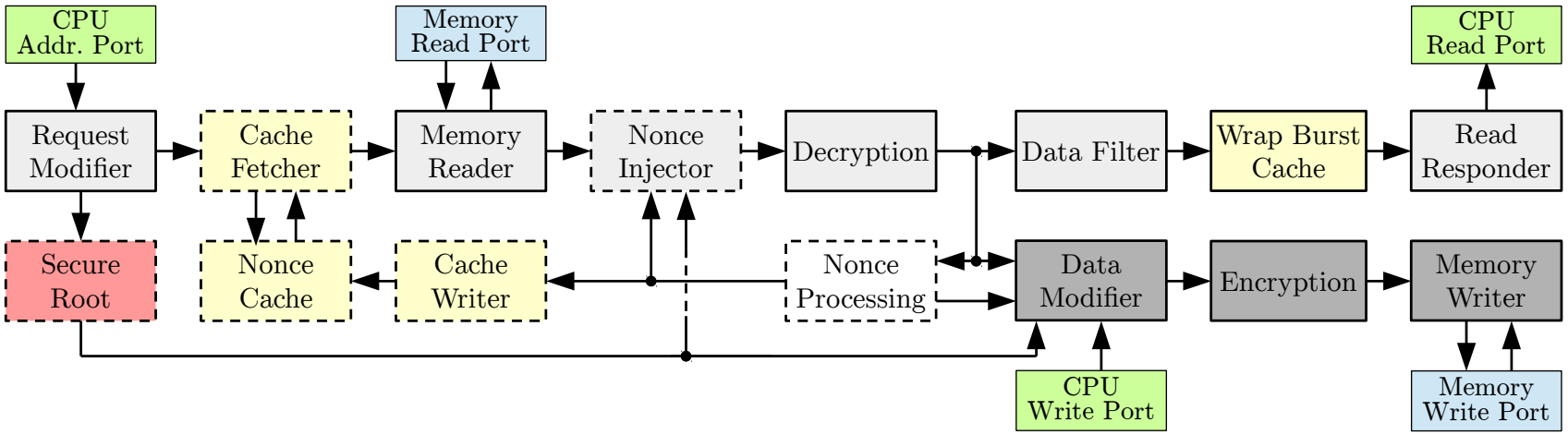


# AXI4 Encryption/Authentication Pipeline



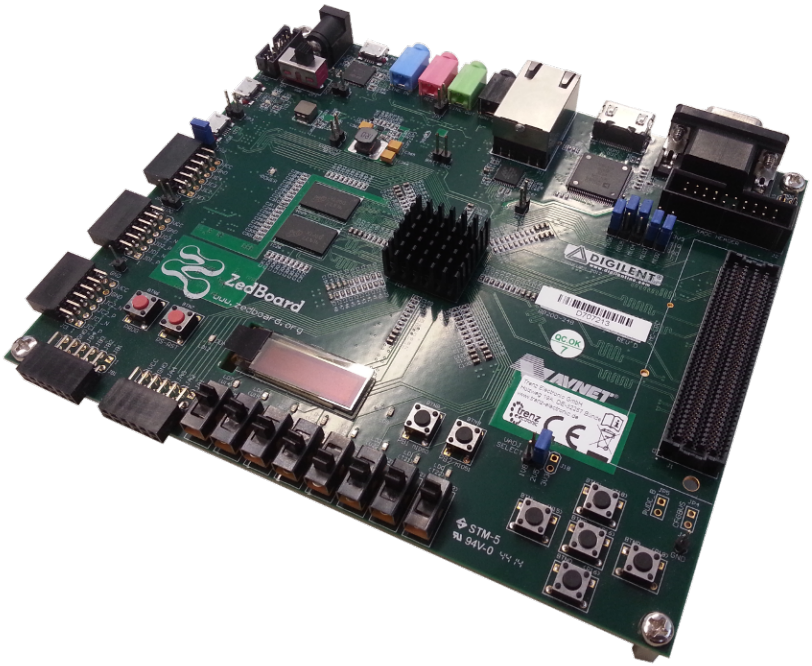

- Writes are always RMW
- Supports block-wise cipher modes (incl. metadata)
- Optimizes wrapping bursts when reading

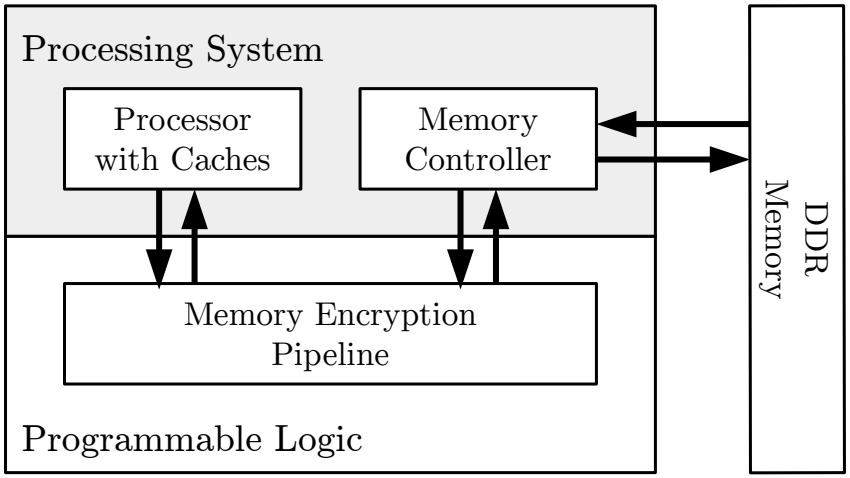
# AXI4 Authentication Tree-Pipeline



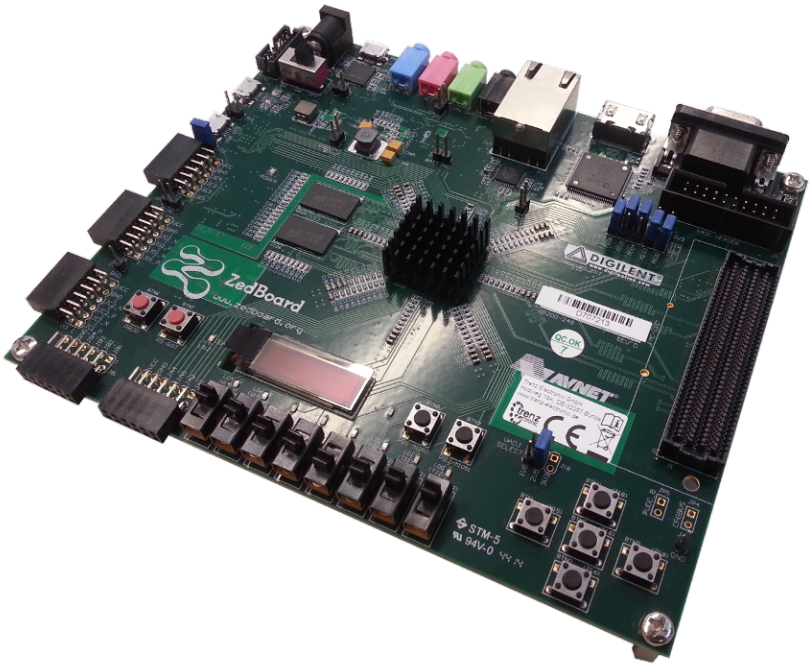

- Writes are always RMW
- Single traversal from the root to the leaf
- Speeding it up: caches for read, parallel trees for write
- Arbitrary tree arity

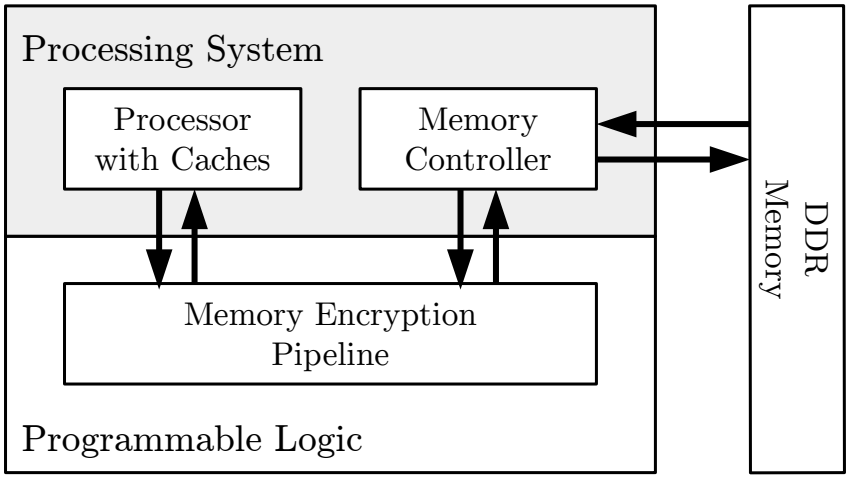
# Evaluation Platform



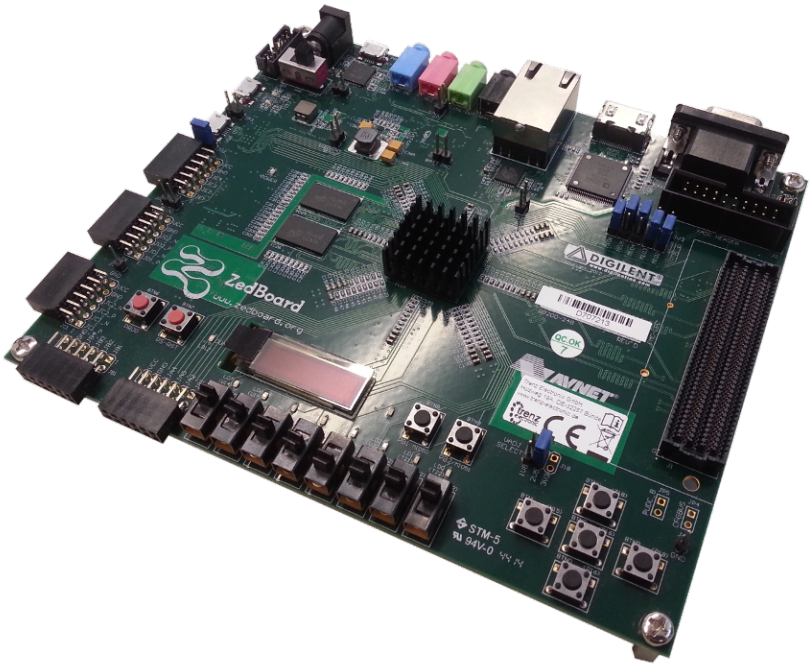
# Evaluation Platform

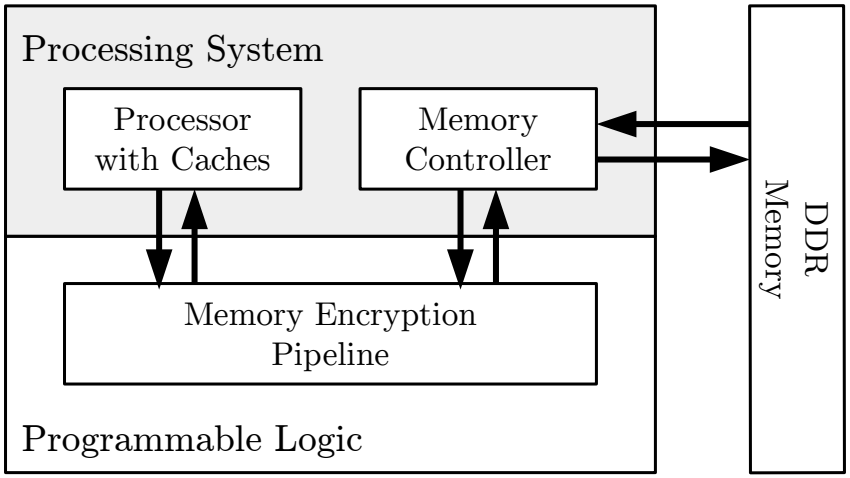



- Block size: cache line

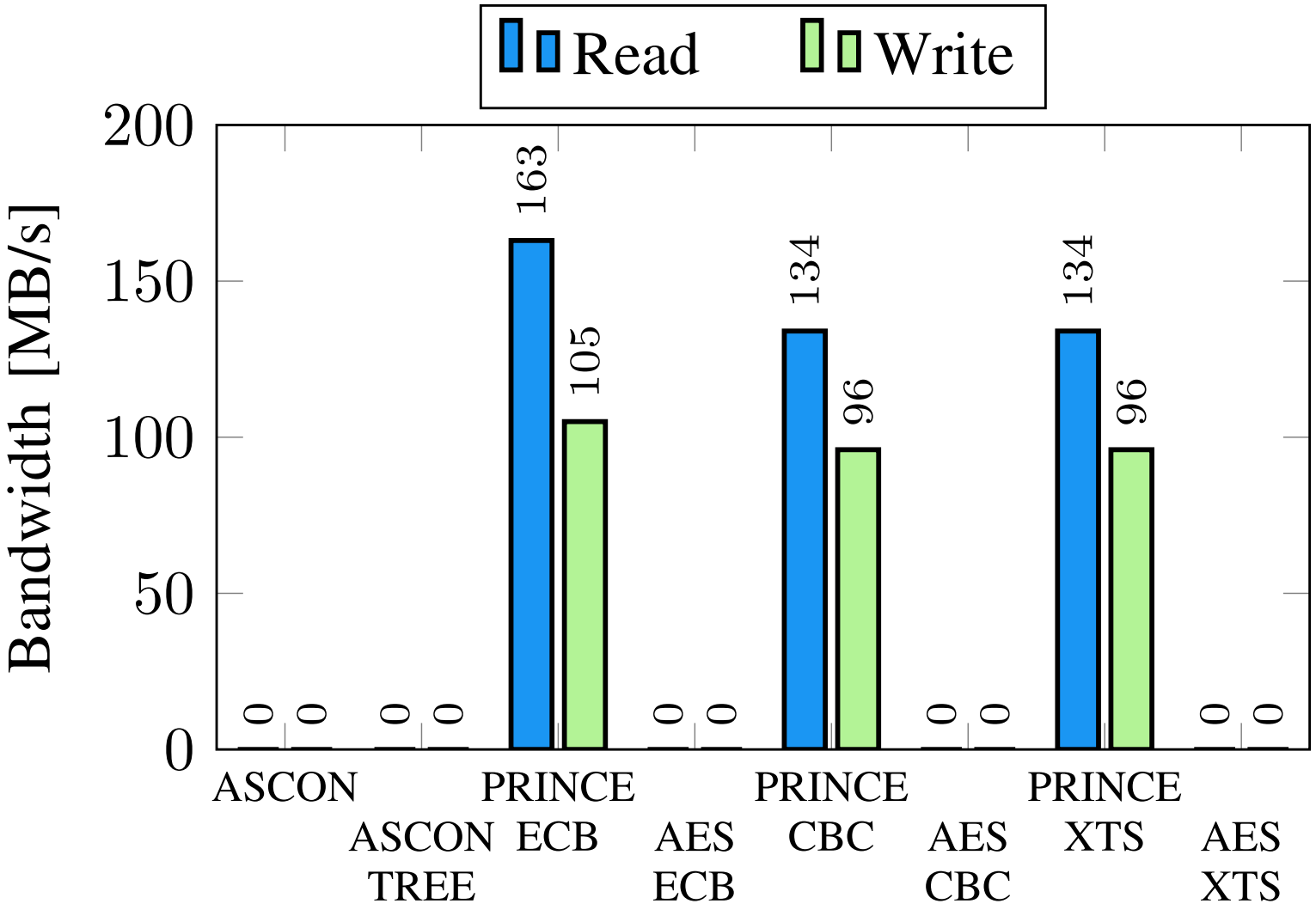
# Evaluation Platform



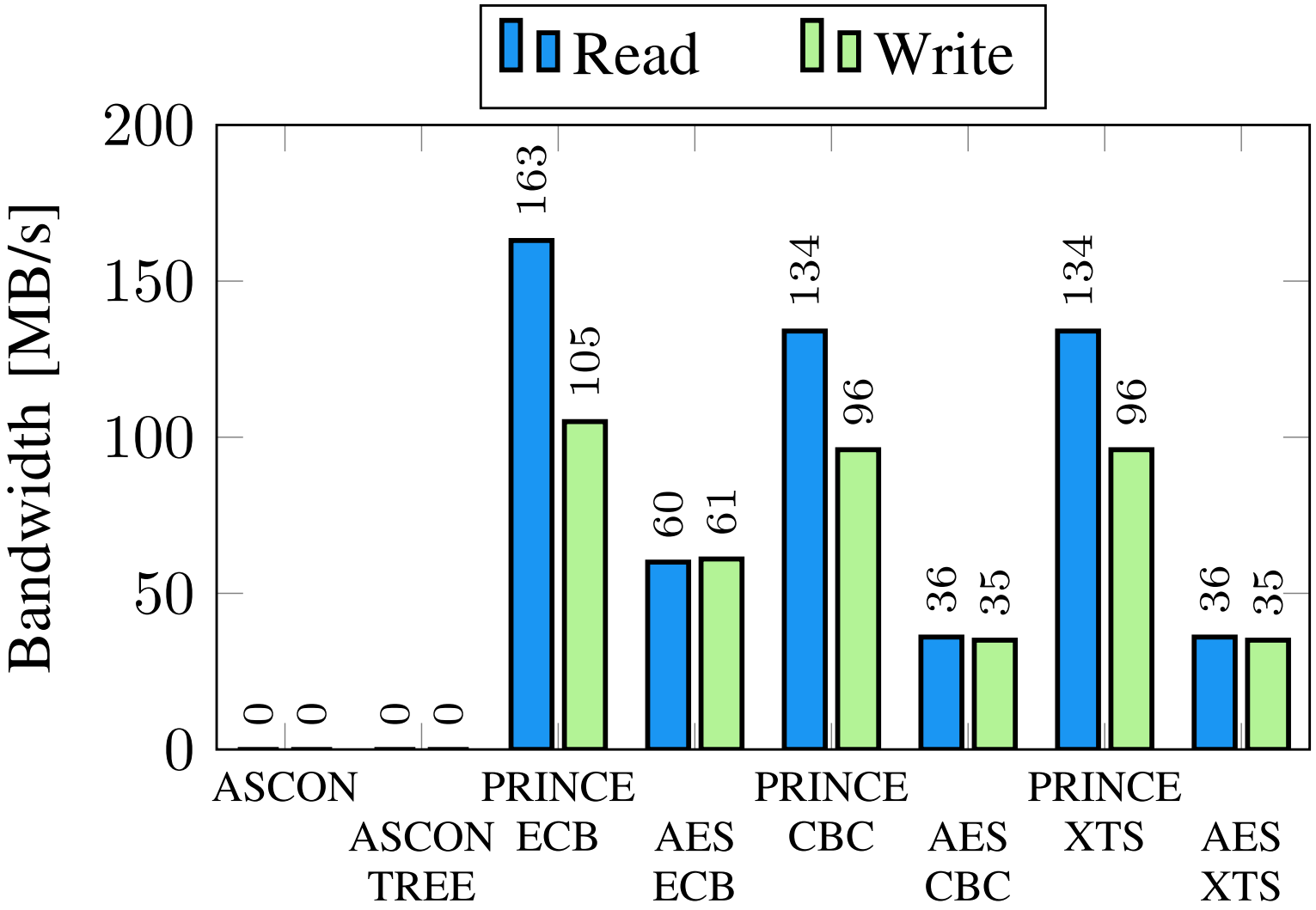




- Block size: cache line
- Running Linux and tinymembench

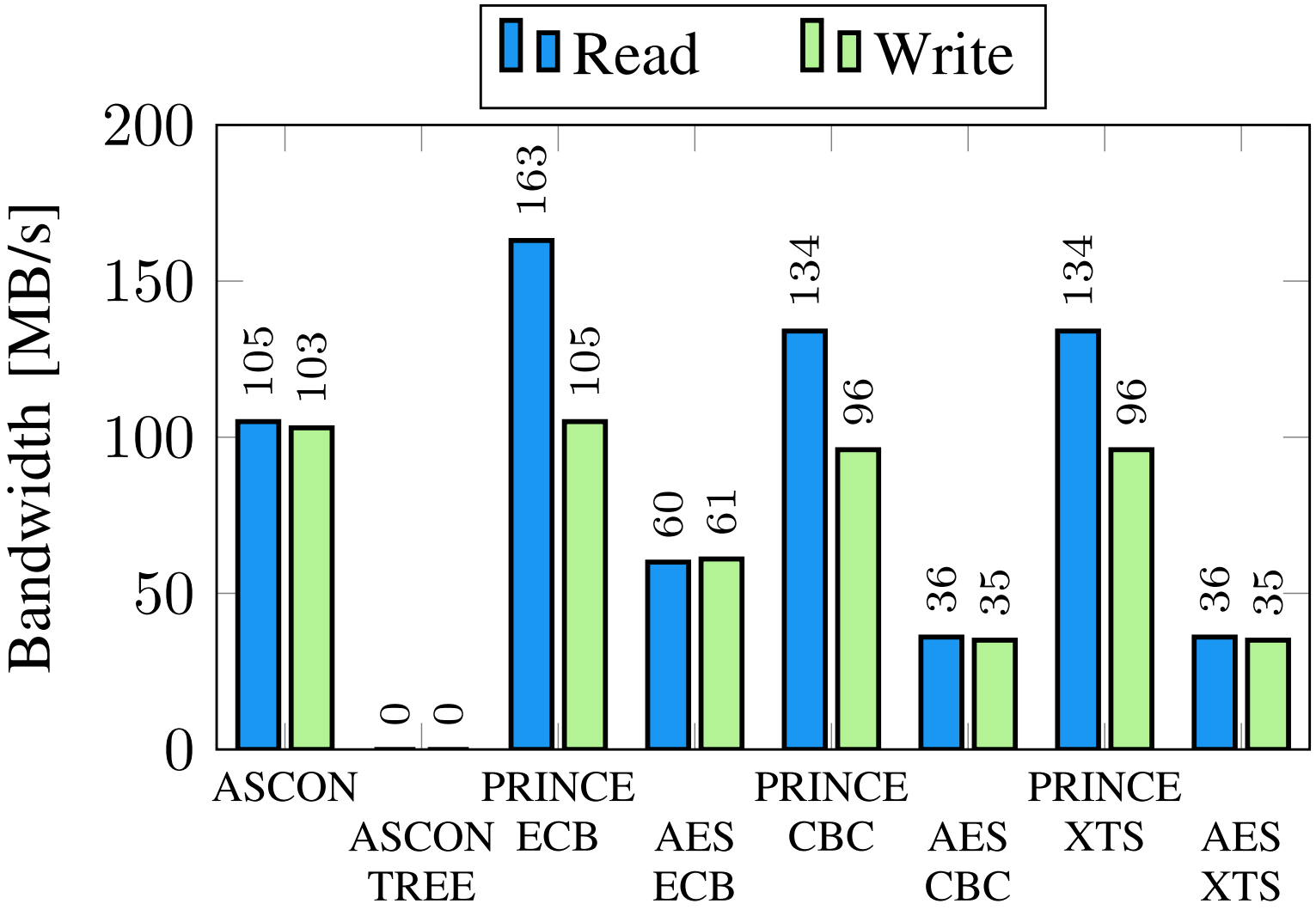
# Bandwidth @ 50 MHz, 200 MB/s max.



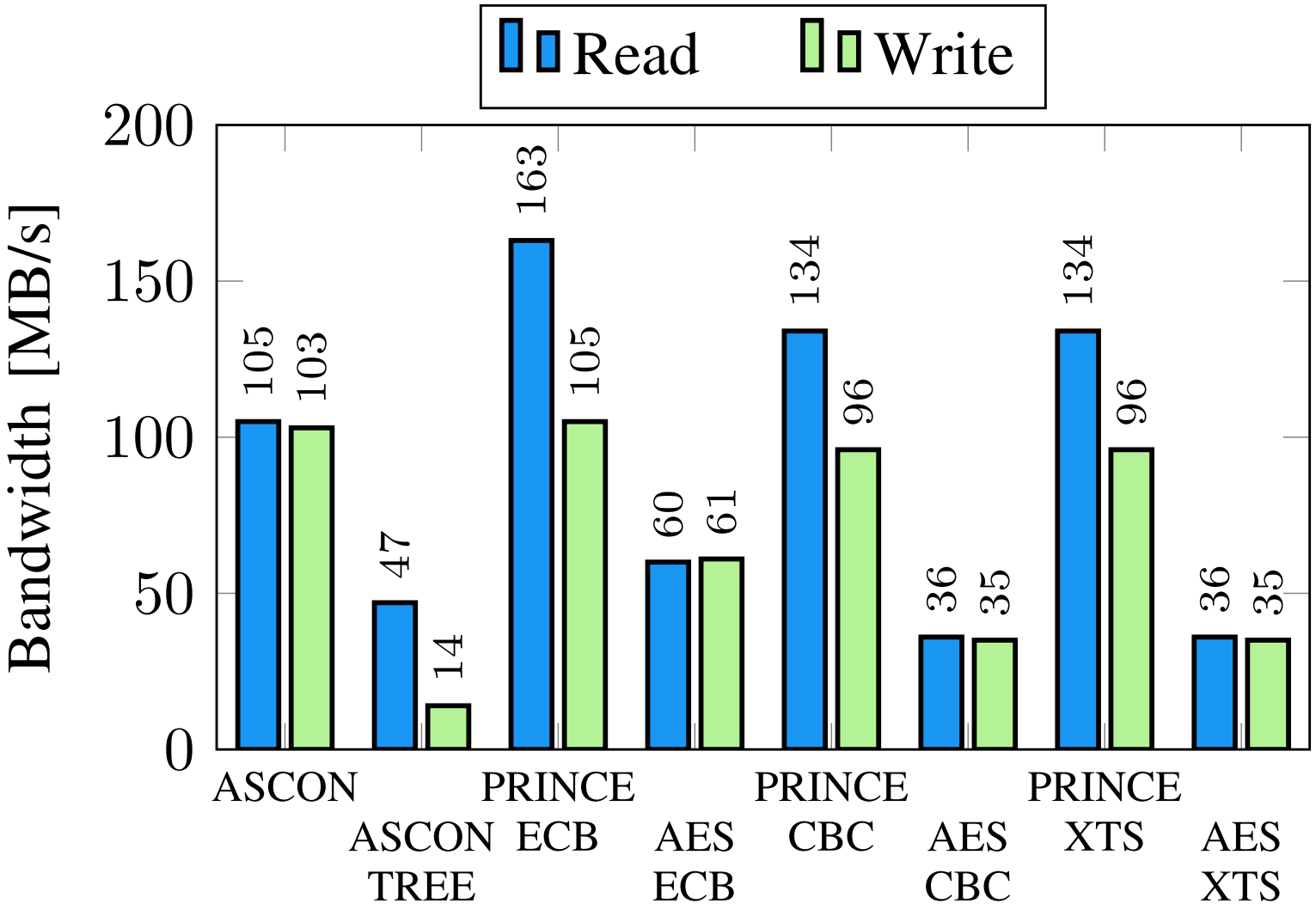
# Bandwidth @ 50 MHz, 200 MB/s max.



# Bandwidth @ 50 MHz, 200 MB/s max.



# Bandwidth @ 50 MHz, 200 MB/s max.



# Conclusion

- Ensure confidentiality and authenticity of data in RAM
- Open-source framework for RAM encryption and authentication
  - Supports various cipher modes: AES/PRINCE, ECB/XTS/CBC, ASCON
  - Tree-based authentication, e.g., ASCON TEC tree
  - Evaluation on Xilinx Zynq-7020 SoC FPGA
  - Efficient pipeline: 94% bandwith utilization
  - ASCON AE for authenticity and efficiency

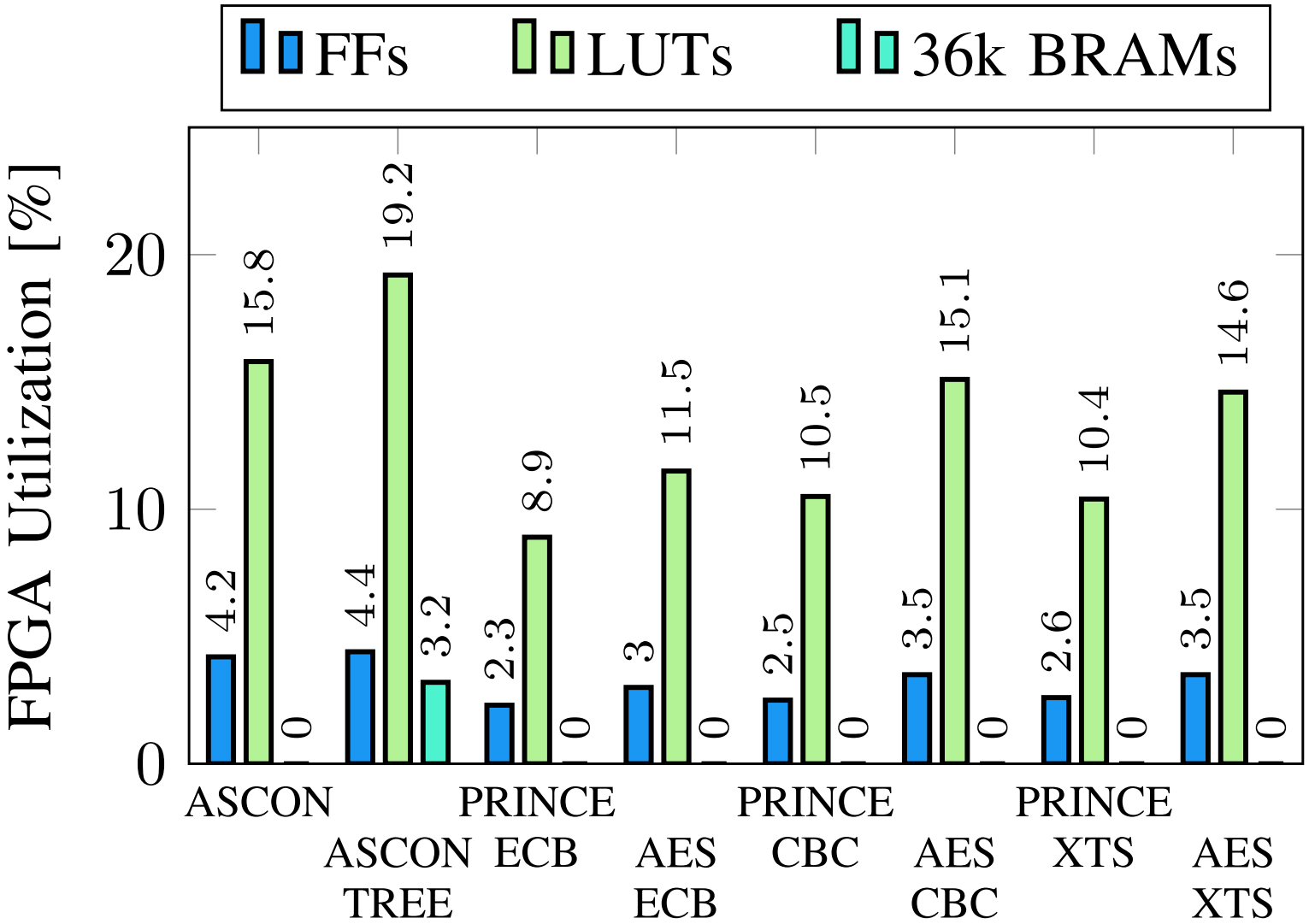
https://github.com/IAIK/memsec

# Transparent Memory Encryption and Authentication

**Mario Werner, *Thomas Unterluggauer*, Robert Schilling, David Schaffenrath, and Stefan Mangard,**
**IAIK, Graz University of Technology**

6. September 2017

# FPGA Utilization (target: 50 MHz)



# Block Size and Bandwidth (PRINCE CBC)