arm

The ARM TARDiS Team & friends proudly present

Memory Protection
for the ARM Architecture

Dr. ARM (Avanzi Roberto-Maria)
January 2020, *Nieuw Amsterdam*

© 2019 Arm Limited


arm
TARDiS
Team for
Analysis,
Research, and
Development
in
Security

Authors and acknowledgements

- Roberto Avanzi — ARM ATG Architect, Munich, Germany
  TARDiS Lead, Memory Protection Lead, Cryptographer-in-Residence
- Subhadeep Banik — School of Comp. and Comm. Sci., EPFL, Switzerland
- Orr Dunkelman — Comp. Sci. Dep., University of Haifa, Israel
- Hector Montaner — Graphcore, Cambridge
  This work was done while the author was with ARM
- Prakash Ramrakhyani — ARM Research, Austin, TX
- Francesco Regazzoni — ALaRI, Università della Svizzera italiana, Switzerland

Supported from the EU Horizon 2020 research and innovation program
CERBERO project (grant agreement number 732105)

- Andreas Sandberg — ARM Research Cambridge

\+ work by/with Stephan Diestelhorst, David Schall, Wendy Elsasser, Gururaj Saileshwar and many others

© 2019 Arm Limited
arm

I will talk about

confidentiality

of memory contents

and memory

integrity violation detection

Why are we talking about protecting memory contents?

- Sensitive assets are in RAM – there is a security problem
- A cat-and-mouse game begins…
- RAM can be read by SW? *Access Control!*
- Cold boot / platform reset attacks? *Encrypt! Ephemeral keys!*
- Attacks that can adaptively *modify* memory contents? *Freshness! Integrity!*
- Commercial and academic solutions abound: AEGIS, SGX, and co. …
- To be clear, I am not announcing any *feature* today: this is not my job
- But we would be a bunch of idiots if we were not studying this
- Here is *what* ***we have studied*** *to protect memory contents cryptographically*

© 2019 Arm Limited
arm

Threat models

An important remark

Memory protection is needed when the owner of some SW or data does not want the internal state of their stuff to leak or be tampered with while it is running and beyond:

A software module running on your own device …

… or your application or VM running in the cloud

In both scenarios, *the SW runs on somebody else's computer!*

In both scenarios, an attacker may use SW running on the same platform or HW manipulation

The HW's owner can be an adversary

So: what is in the security perimeter and what is not?

The CPU, its bootROM are inside, what else?

© 2019 Arm Limited

There are two scenarios (or three)

- **Either the memory is “internal”**, as in
  - On the same die with the CPU
  - PIP: In the same package as the CPU masters (+ anti tamper?)

  Cannot interpose, or at least very very difficult

  Assume memory device trusted

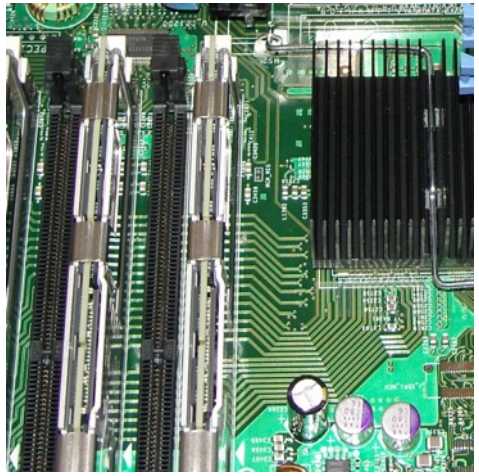
  Threats: Cold boot / platform reset (Rowhammer & co., Fault injection?)

- **Or it is “external”**, as in
  - Socketed
  - Soldered on motherboard
  - POP: Package on package

  Interposing not that difficult

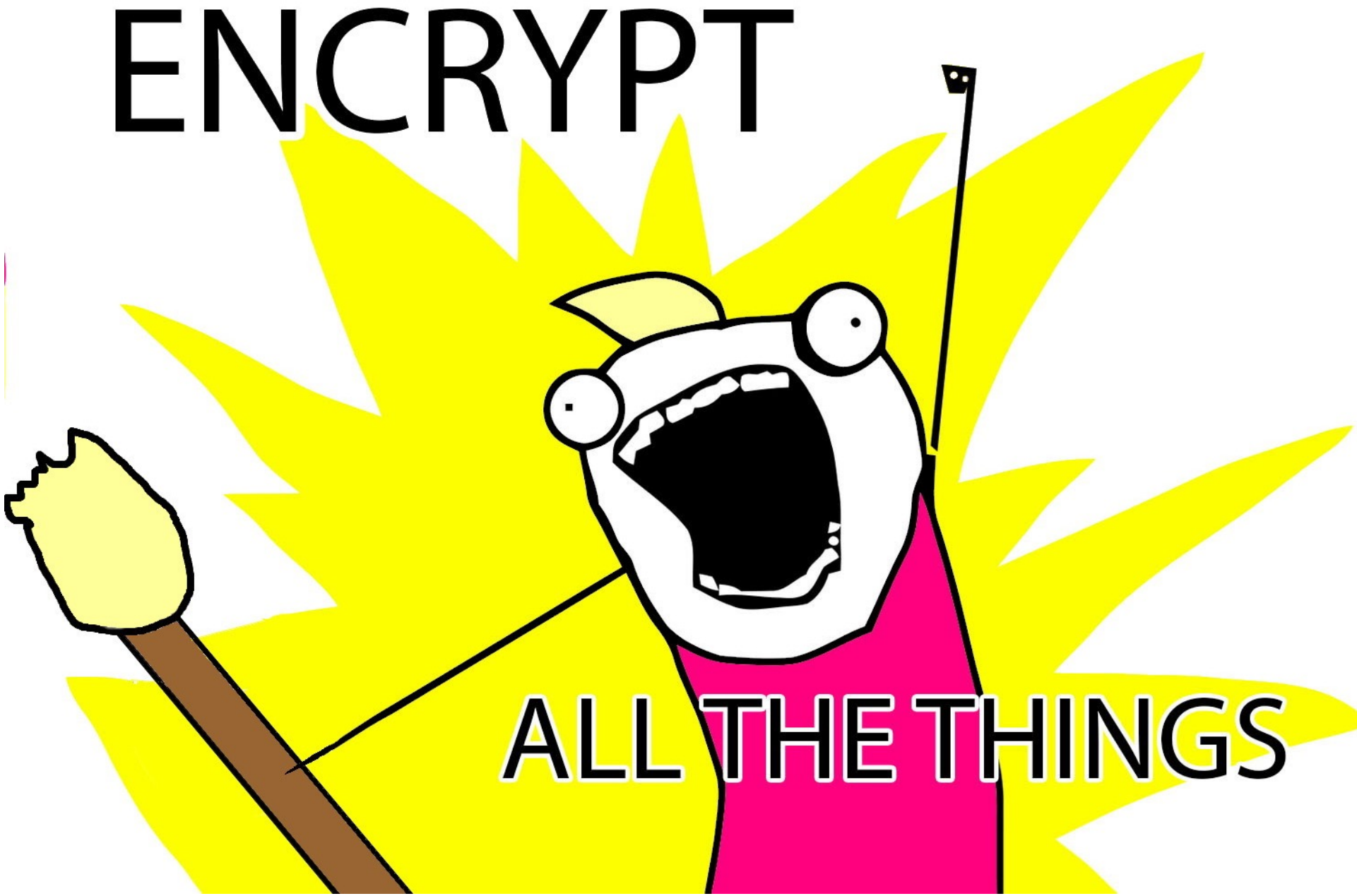
  Assume memory untrusted

  Threats += bus reading/tampering…

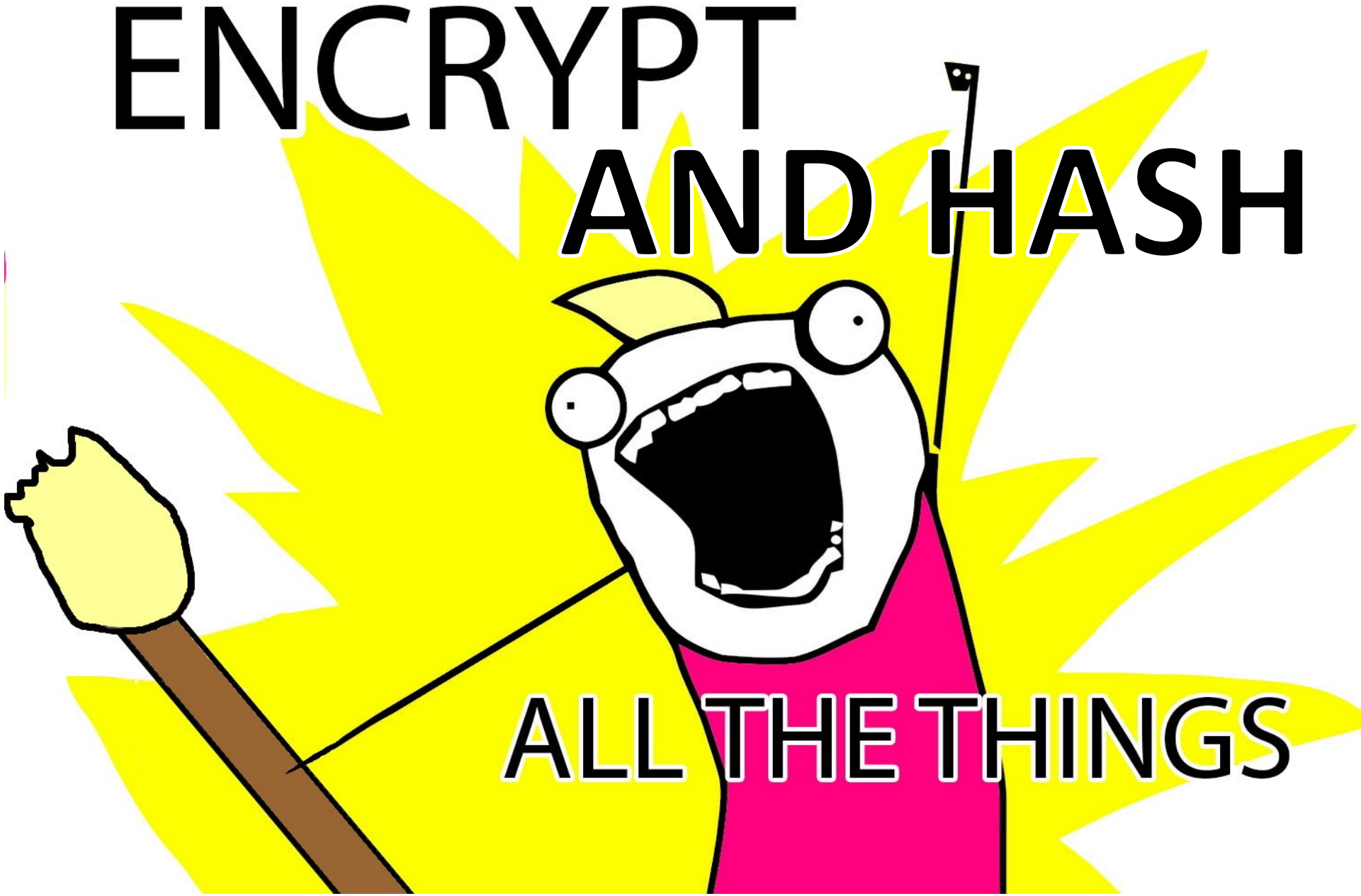


© 2019 Arm Limited
arm

Easy solutions:
we already mentioned them



© 2019 Arm Limited
arm



© 2019 Arm Limited
arm

© 2019 Arm Limited
arm

arm

Done!

Are we?

No, actually.

Cost


BANQUE DU CANADA
CANADA
BANK OF CANADA
THIS NOTE IS LEGAL TENDER
CE BILLET A COURS LÉGAL
GOVERNOR GOUVERNEUR

There are two things that nobody expects:

1. … the Spanish Inquisition

2. … that any piece of technology that kills performance and "wastes" memory for whatever purpose will ever be widely deployed – unless it is spy/ad-ware or bloatware (including blockchains ;-)

© 2019 Arm Limited

- SGX: sound cryptographic protection but 26.7% memory overhead and 25% performance penalty
- AMD-SEV: on paper better performance, no memory overheads, but encryption is not nonced and there is no integrity

© 2019 Arm Limited

Project **MOPED** @ ARM

Memory Opacification: Performance Evaluation and Design

SotA survey
new ideas
benchmarking
selection

Primitives, Key Lengths, Schemes

- **Confidentiality**
  - "Top secret" security level against various adversaries
  - 20 years of classical security minimum ("worst" of `www.keylength.com`)
  - *Add adequate PQ resistance at smallest possible cost*
    - *≥ 128 bits of security (both classical and PQ) with 256-bit keys (because PQ)*

- **Integrity**
  - It shall be *computationally infeasible* to corrupt memory by forging a hash/MAC
  - Same time restrictions on data observation/collection as for confidentiality
  - ≥ 60 bit MACs, include ≥ 64 bit counters in their computation

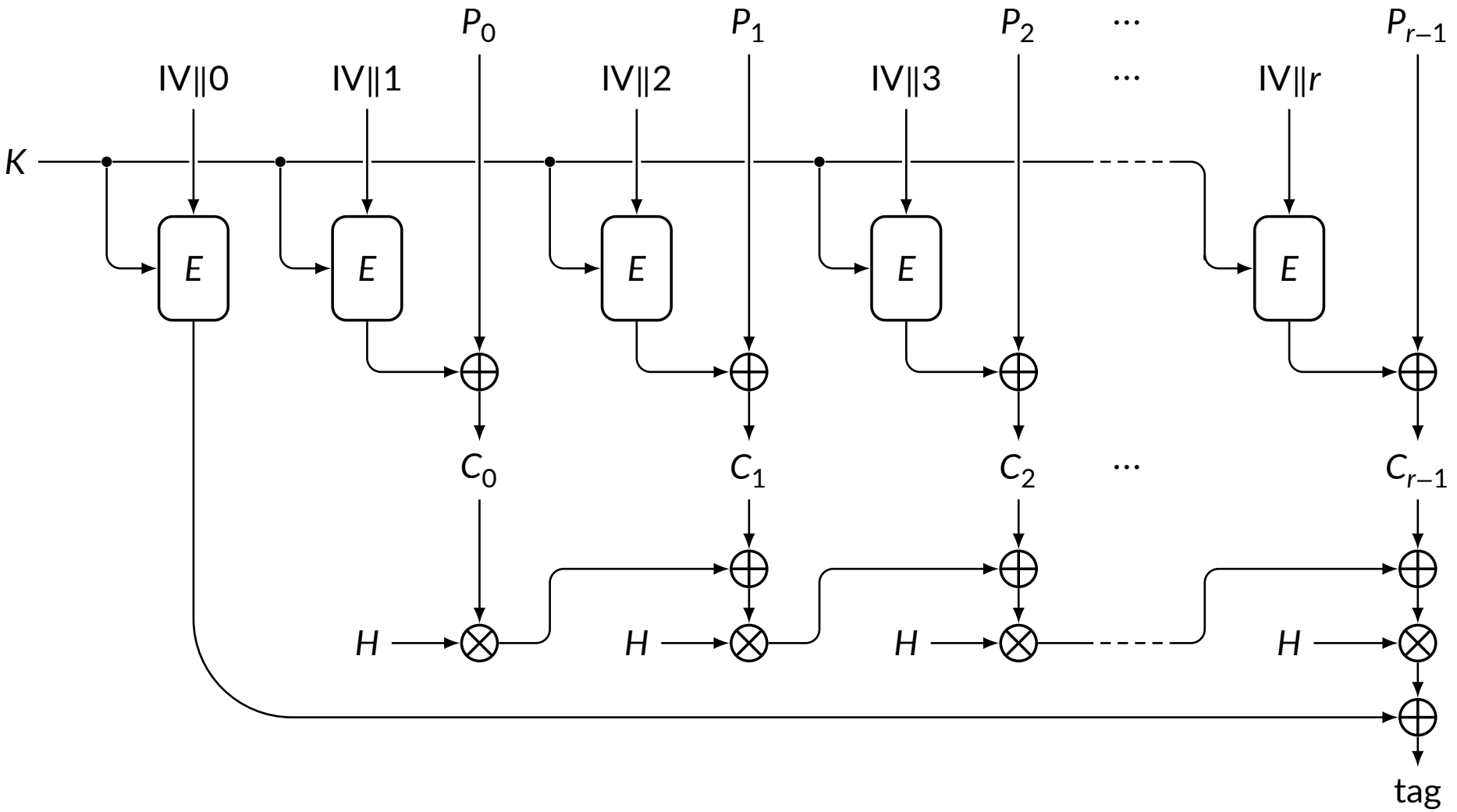
© 2019 Arm Limited

Extensive SotA review: Encryption

- **Various primitives evaluated: the survivors are**
  - **AES** (“standard”, but not really, and very expensive)
  - **Deoxys** (the AES turned into a so-called *tweakable* cipher in a smart way)
  - **QARMA** (developed for “right size” – PD, solid theory, well analyzed)
- **Various modes of operation**
  - Direct encryption and OTP encryption
  - Various types of hashing / MAC algorithms
- **A few examples will follow**
  - Empathize with us – we implemented all of them and many more

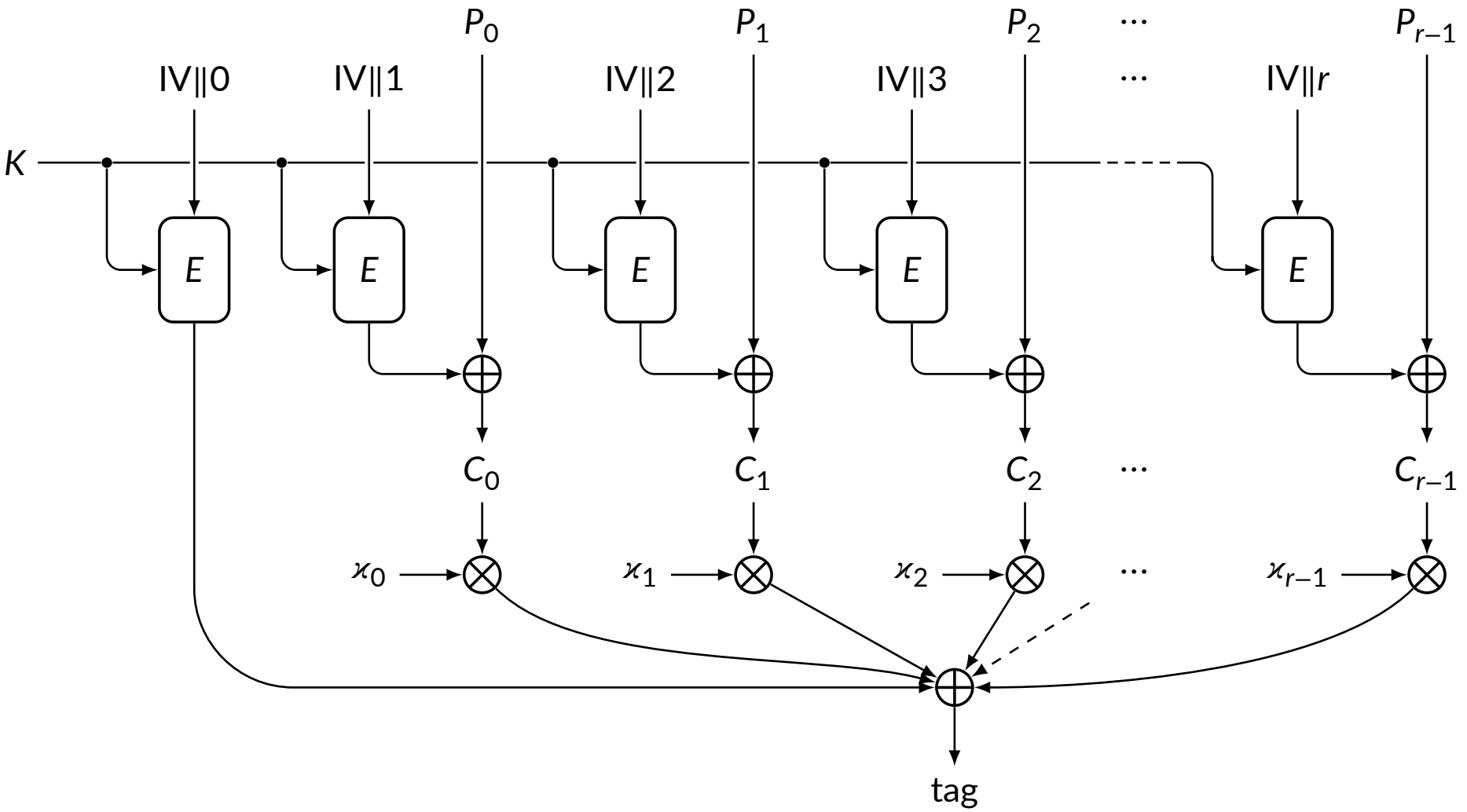
© 2019 Arm Limited

GCM



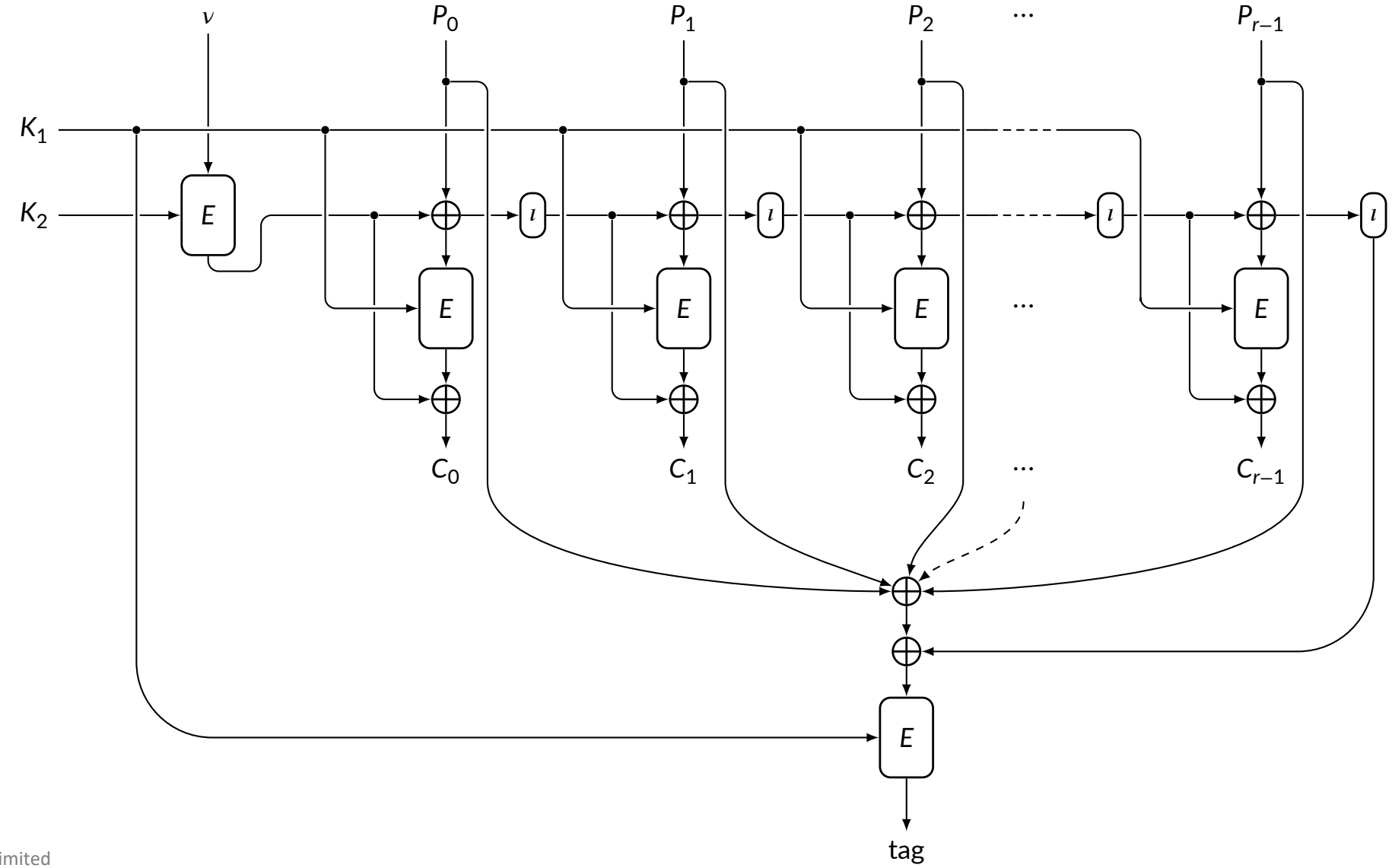
© 2019 Arm Limited
arm

CTR mode with Encrypted Multilinear UHF as Authenticator



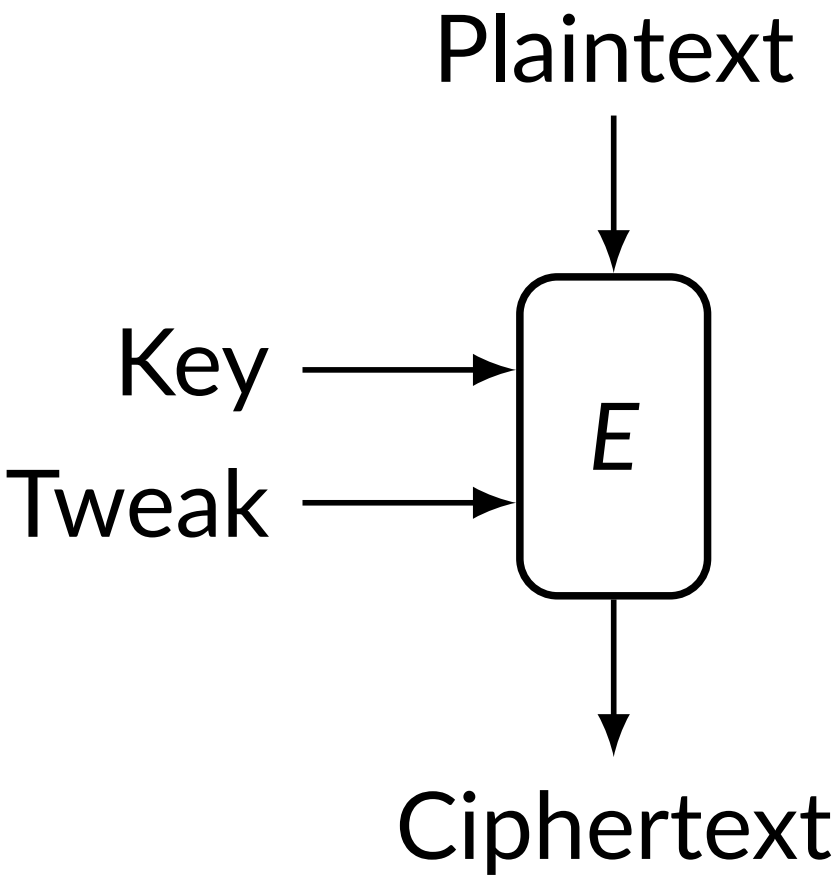
© 2019 Arm Limited
arm

pre-OCB with XEX instantiation



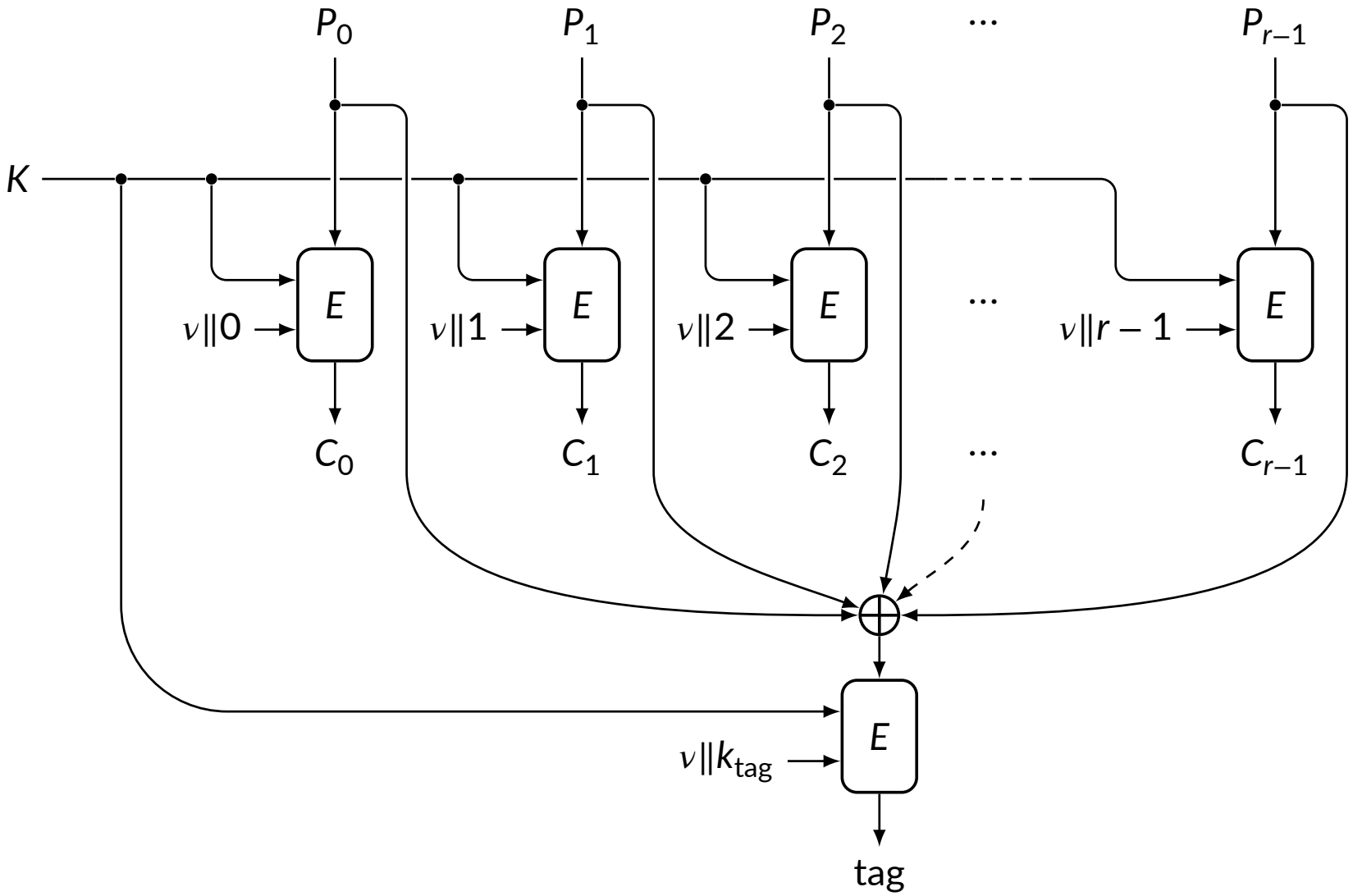
© 2019 Arm Limited

Using a Tweakable Cipher



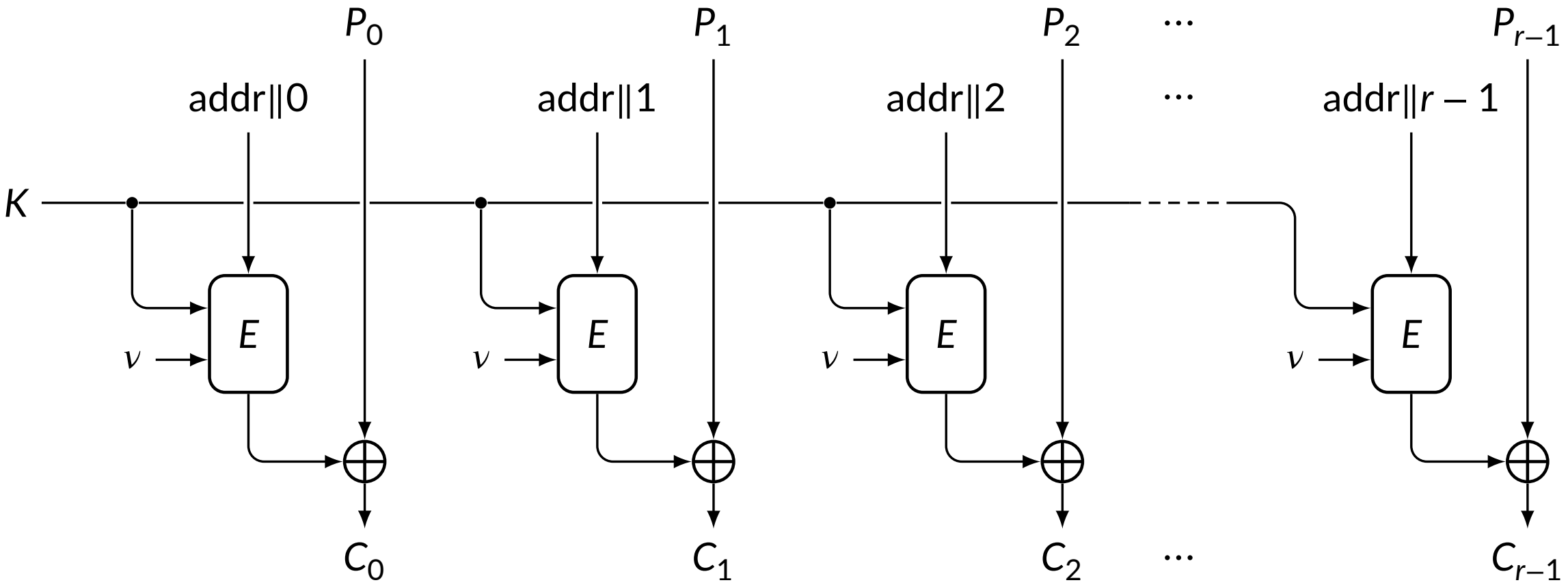
© 2019 Arm Limited
arm

ΘCB–like mode



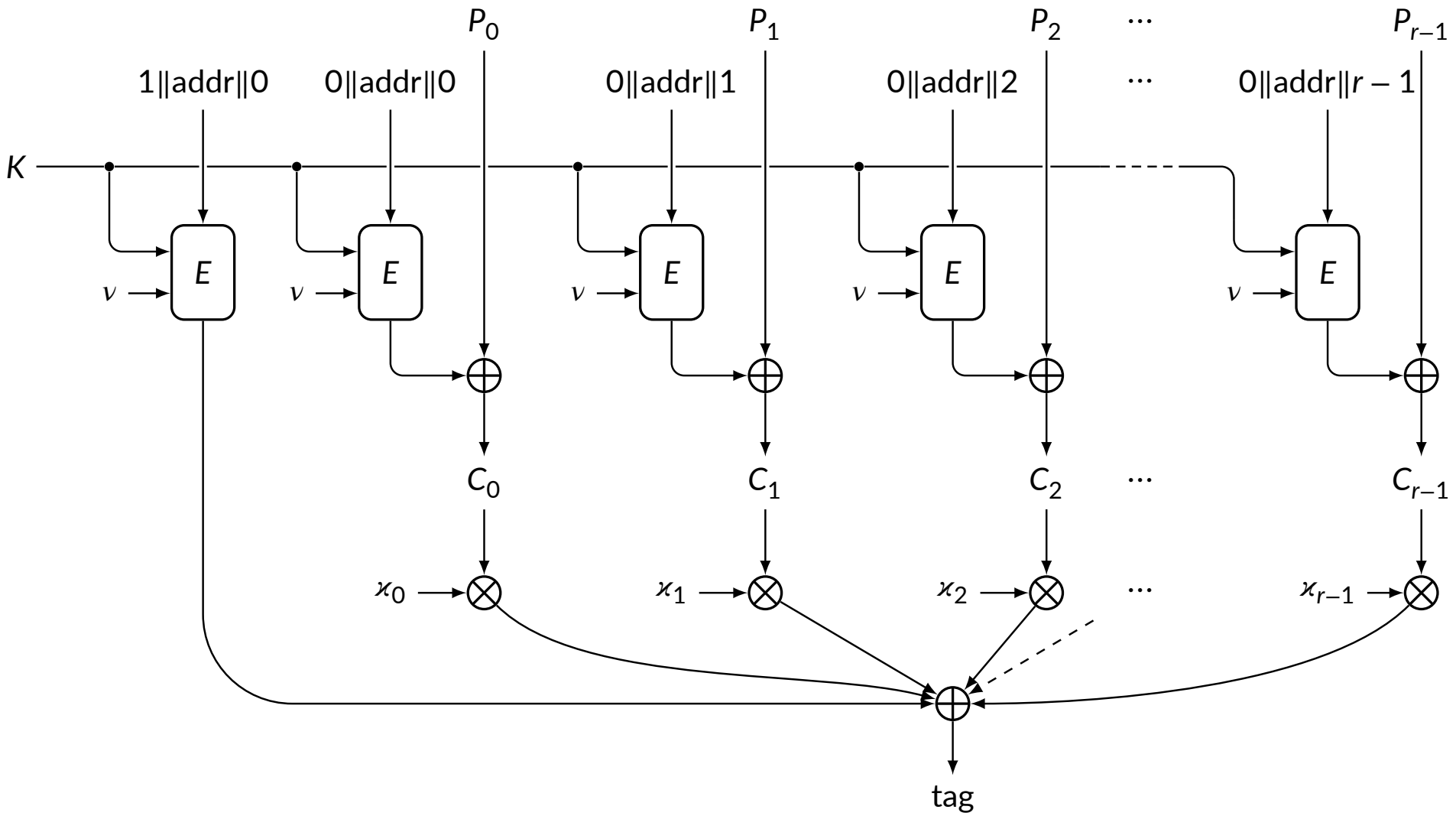
© 2019 Arm Limited
arm

“Counter-in-Tweak” Encryption



© 2019 Arm Limited
arm

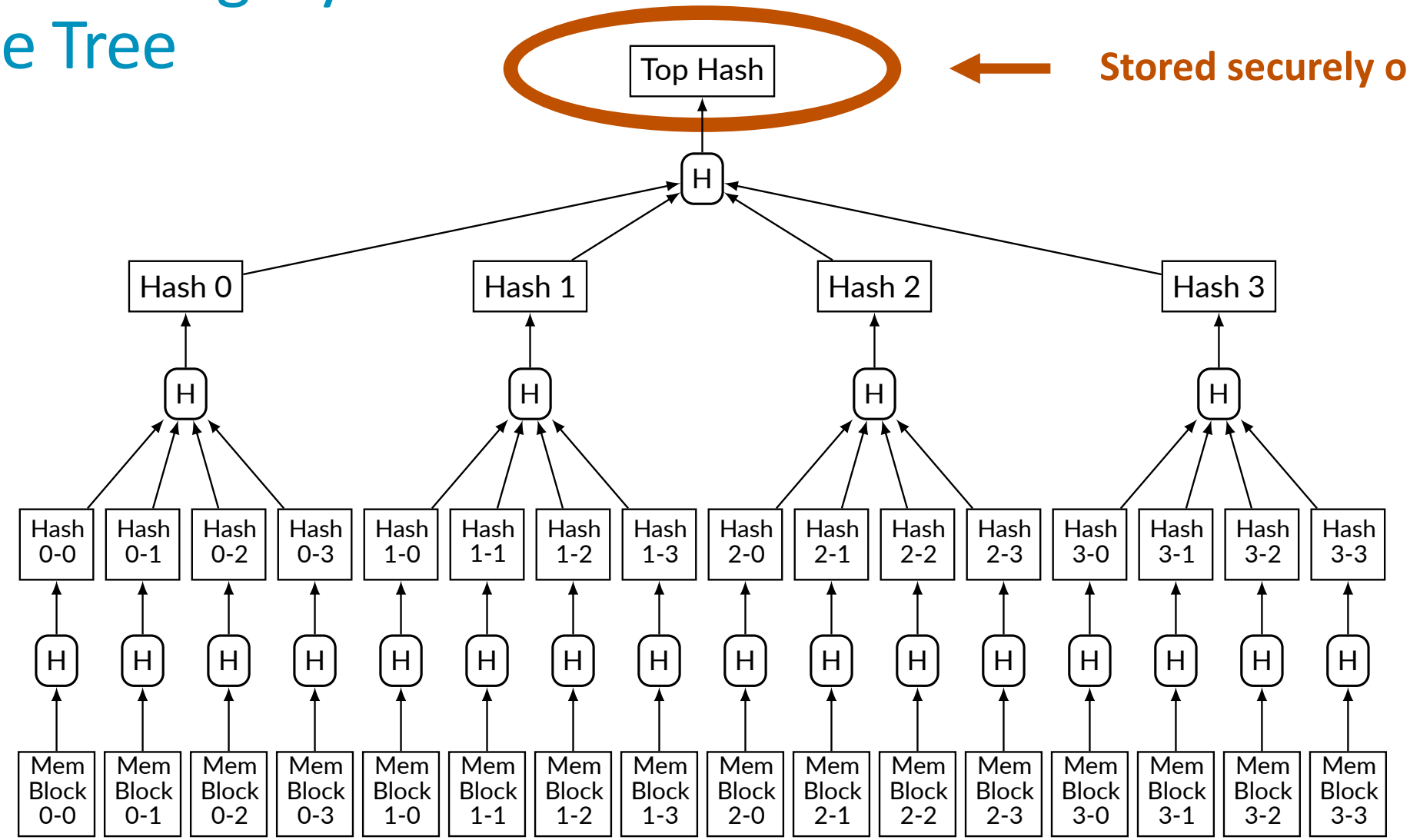
Counter-in-Tweak with Multilinear Hash for AE



© 2019 Arm Limited
arm

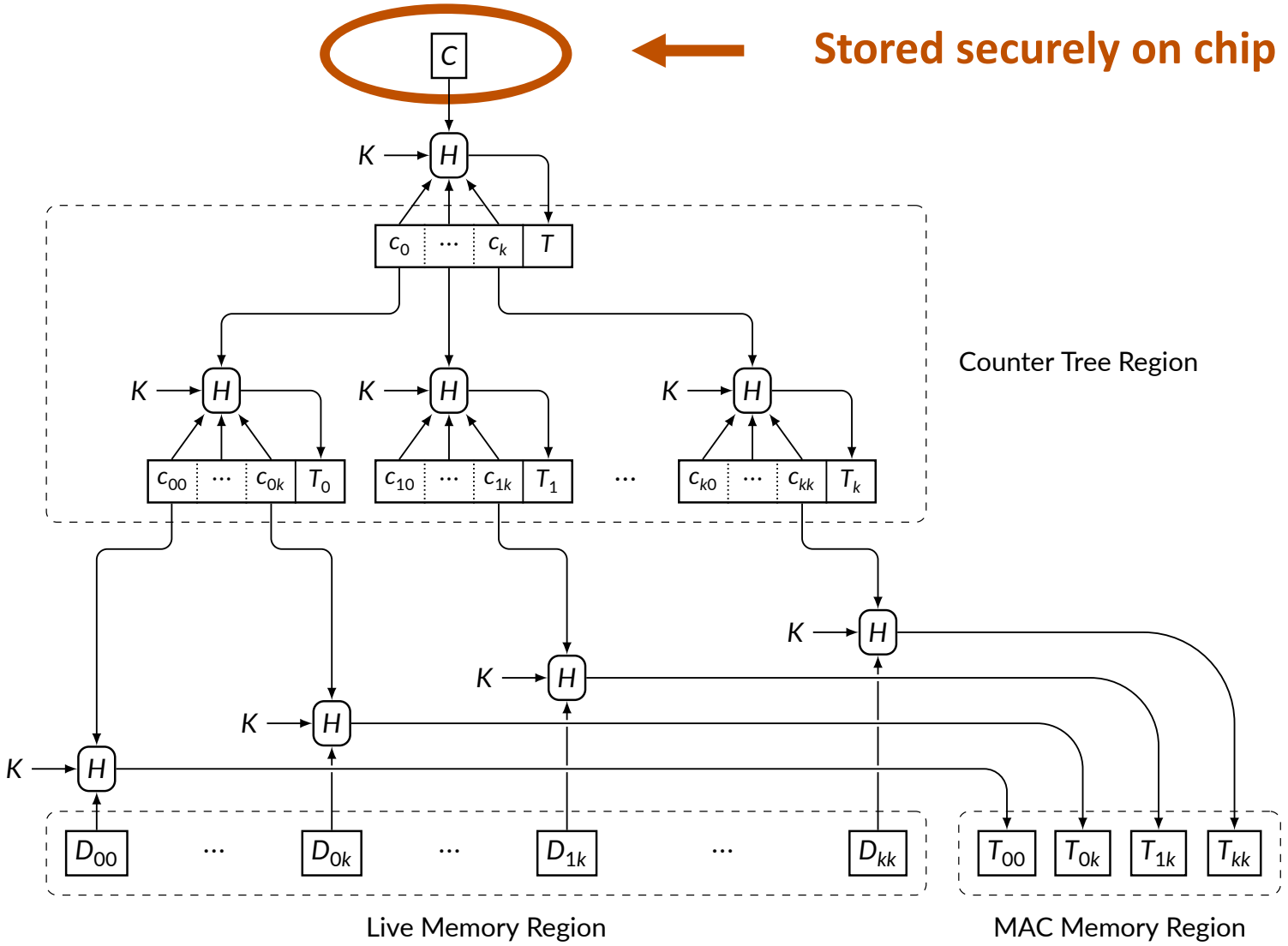
And this was just the *easy* half of the story

And now integrity: Merkle Tree



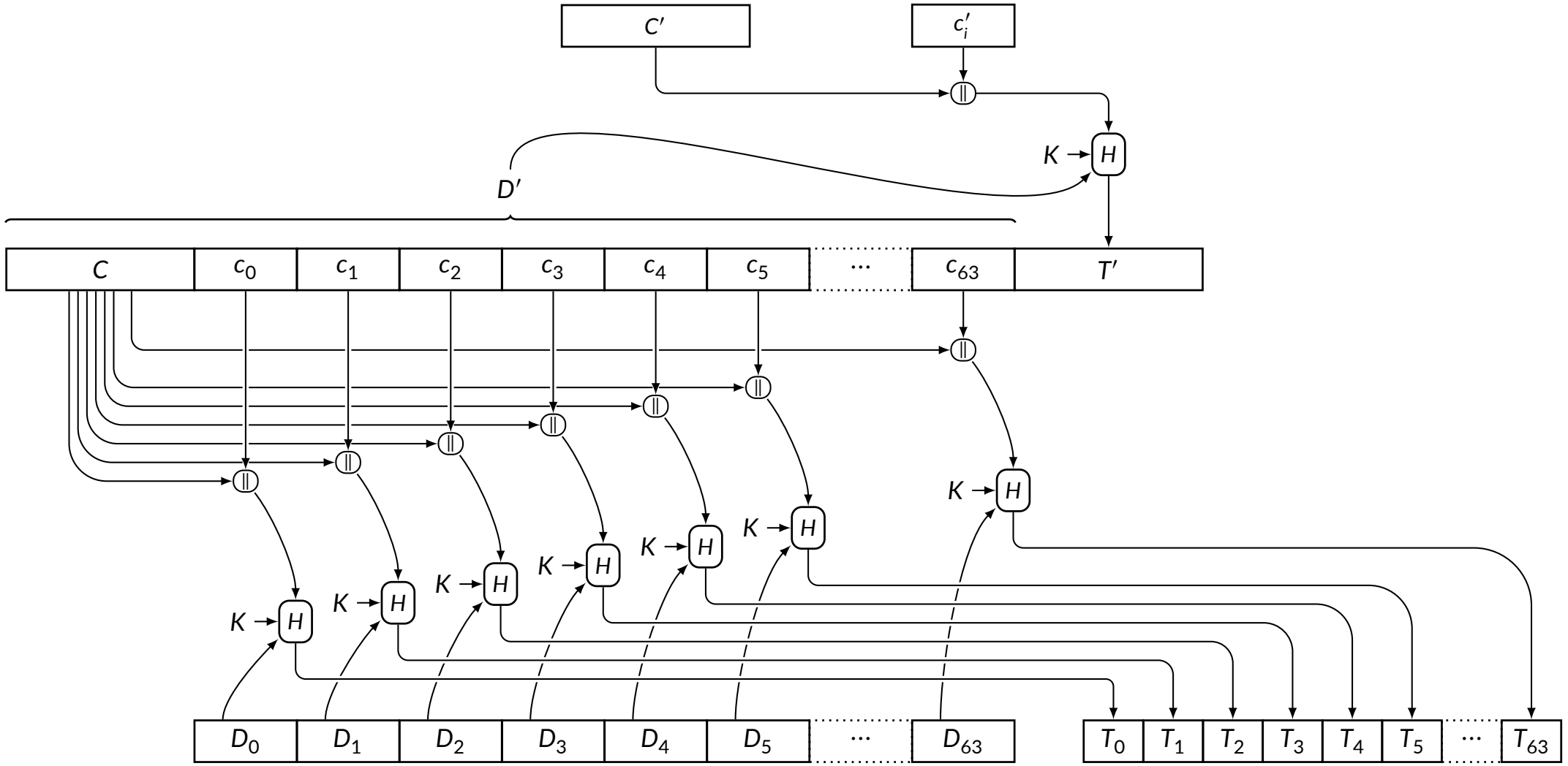
© 2019 Arm Limited
arm

Counter Tree



© 2019 Arm Limited

Split Counters



© 2019 Arm Limited
arm

And I haven’t even talked about variations on the theme of split counters …

TL; DR:
too many variants

Variants

Base methods

0. Nothing
1. Encryption only
2. Encryption and an integrity tag
3. Like 2. + anti-replay counter tree

Encryption variants

A. Direct encryption (no CT freshness)
   [AES/XEX or a TBC]
B. OTP encryption (with CT freshness)
   [AES/CTR or TBC/CIT]

Counter variants

m. Monolithic
s. Split

MAC variants

i. One MAC per CL
ii. One MAC per 2 CLs
iii. One MAC per 4 CLs (and so on?)

Verify MACs synchronously or asynch

We cannot realistically compare all variants, so we select some

© 2019 Arm Limited

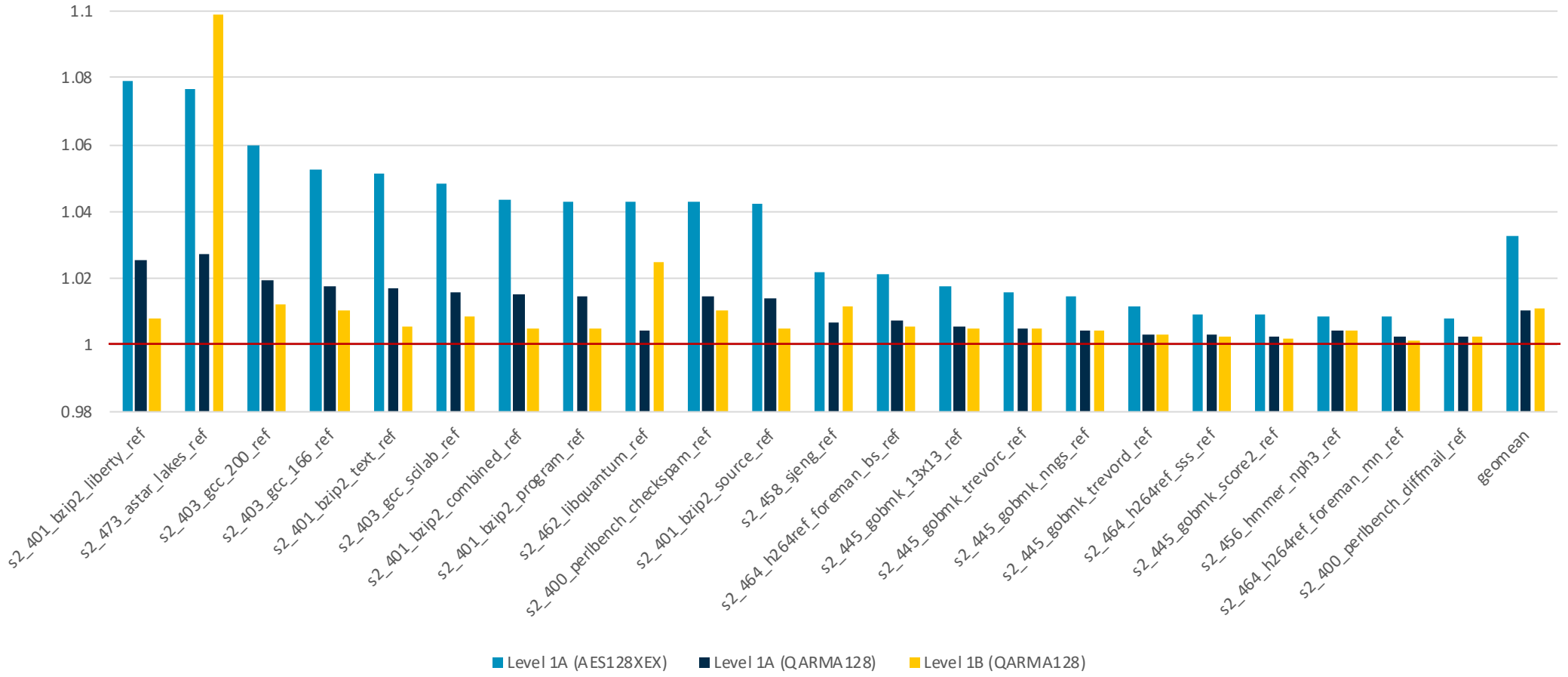

Competition

Benchmarks

- We run SPEC (int) 2006 & 2017 on full system emulators
- They target the A57 and A75 cores
- The A57 simulator works perfectly
- Some benchmarks do not yet run on the A75 one
- The crypto HW is emulated by inserting latencies in the data paths, obtained by synthesis
- Memory is 8Gb, DDR4_2400_4x16, 1 channel

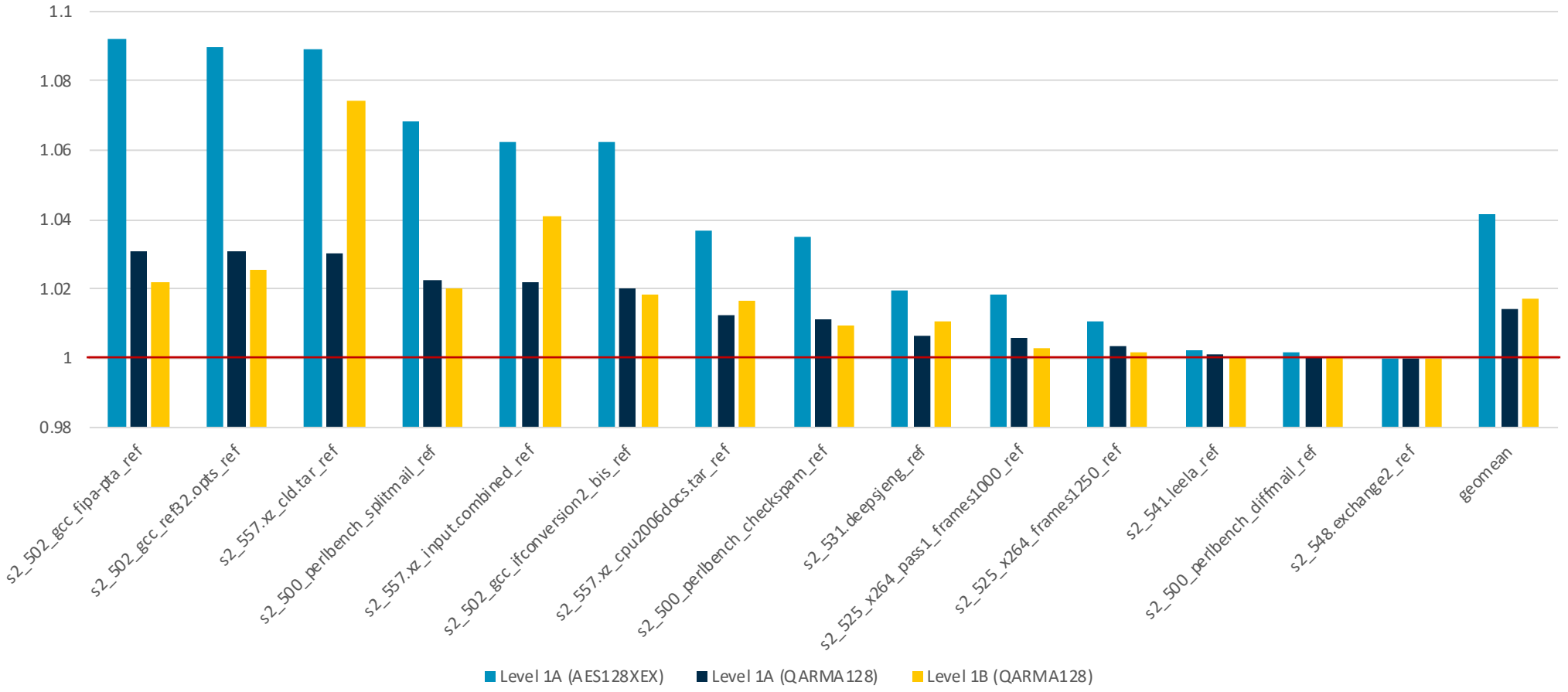
© 2019 Arm Limited

Level 1: Encryption only on SPEC 2006



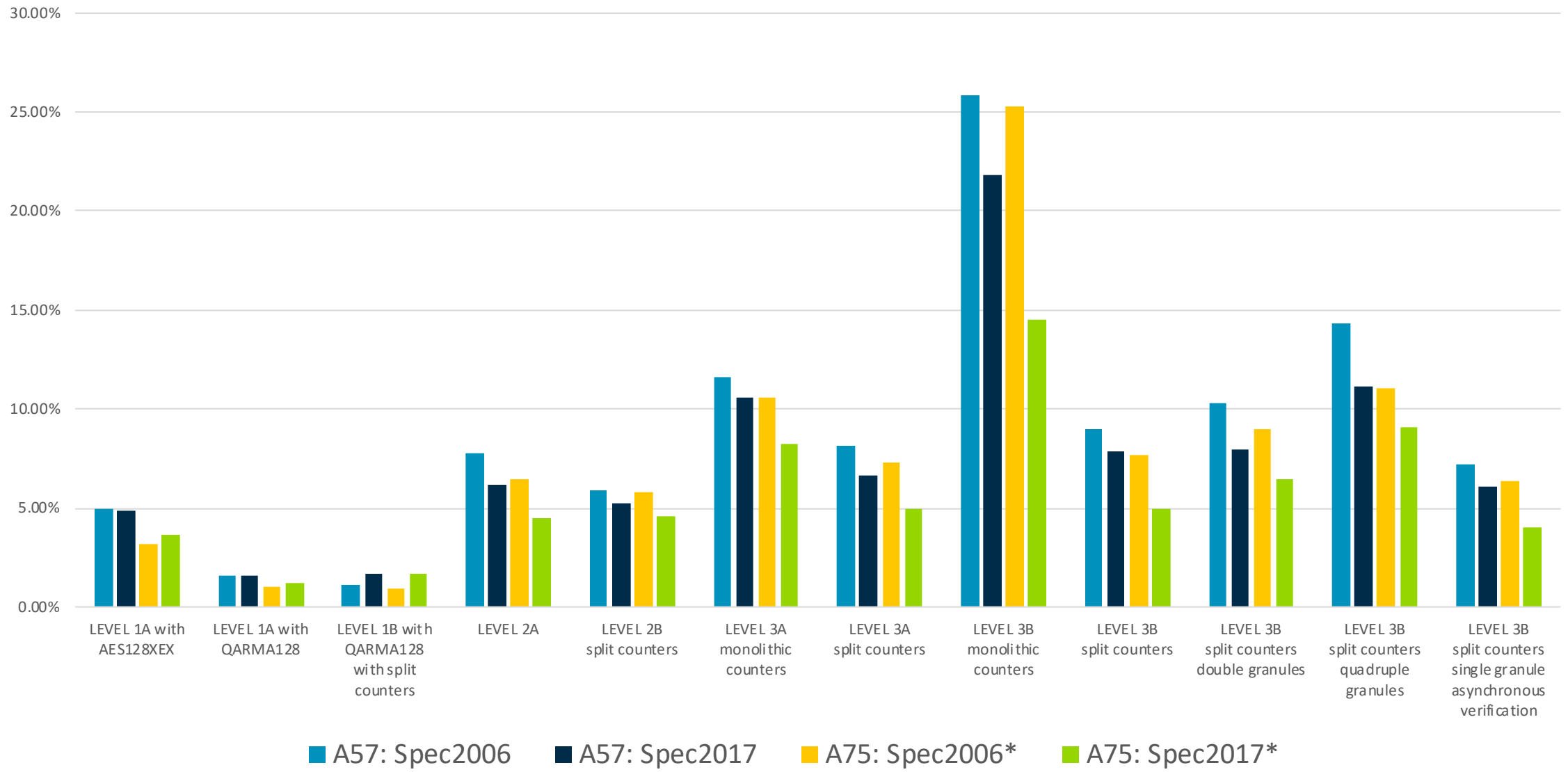
© 2019 Arm Limited

Level 1: Encryption only on SPEC 2017



© 2019 Arm Limited
arm

Memory protection techniques: Performance penalty



© 2019 Arm Limited

Memory Overheads

Structure		Integrity granule size: 512 bits	1024 bits	2048 bits
	Merkle Tree with 128-bit hashes	33.3%	14.3%	6.7%
	Merkle Tree with 128-bit hashes and 64-bit freshness counters	45.8%	26.8%	19.2%
	8-ary counter tree (intel) with 56-bit MACs and counters	26.7%	12.9%	6.5%
	Split counter tree with 64-bit MACs with 64+6-bit counters	14.09%	7.84%	4.71%
	Encryption (split 64+6 counters) and integrity	14.06%	7.81%	4.69%
	Encryption only with monolithic 64-bit counters	12.5% (integrity granule independent)		
	Encryption only with split 64+6-bit counters	1.6% (integrity granule independent)		

© 2019 Arm Limited
arm

Memory Overheads

Structure		Integrity granule size: 512 bits	1024 bits	2048 bits
	Merkle Tree with 128-bit hashes	33.3%	14.3%	6.7%
	Merkle Tree with 128-bit hashes and 64-bit freshness counters	45.8%	26.8%	19.2%
	8-ary counter tree (intel) with 56-bit MACs and counters	**26.7%**	12.9%	6.5%
	Split counter tree with 64-bit MACs with 64+6-bit counters	14.09%	**7.84%**	4.71%
	Encryption (split 64+6 counters) and integrity	14.06%	7.81%	4.69%
	Encryption only with monolithic 64-bit counters	12.5% (integrity granule independent)		
	Encryption only with split 64+6-bit counters	1.6% (integrity granule independent)		

© 2019 Arm Limited
arm

Takeaways

Conclusions and additional information

- First comprehensive comparison in this field
- Significant improvements showed wrt SotA: For encryption with integrity and anti-replay from
  - 25% performance penalty and 26.7% memory overhead (SGX) to
  - 7.5% performance penalty and 14.1% memory overhead or even
  - 8.7% performance penalty and 7.8% memory overhead
- Sacrificing freshness or anti-replay does not give a big performance or memory overhead improvement, so it is better to use the whole Santa Barbara
- Encryption only with no freshness is secure only under a non-realistic threat model and costs only a little less than providing full protection – and we can still improve
- Memory bandwidth plateaus to about 183% with the highest protection :-(

© 2019 Arm Limited

Can you do better?

Can you do better?

Then schlep your tuchis to your work desk and show us what you can do … at RWC ’21 in (old) A’dam!!!

arm

Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
תודה

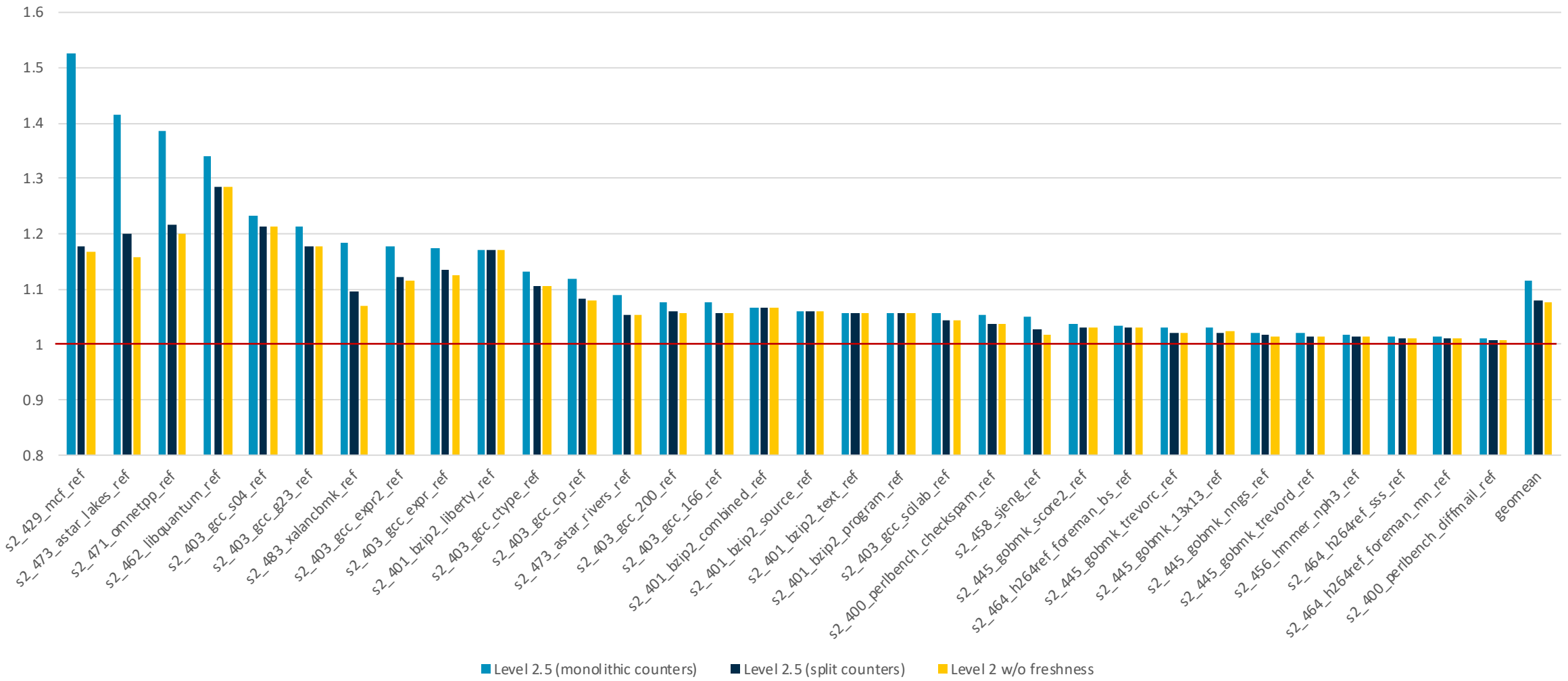
© 2019 Arm Limited

The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

www.arm.com/company/policies/trademarks

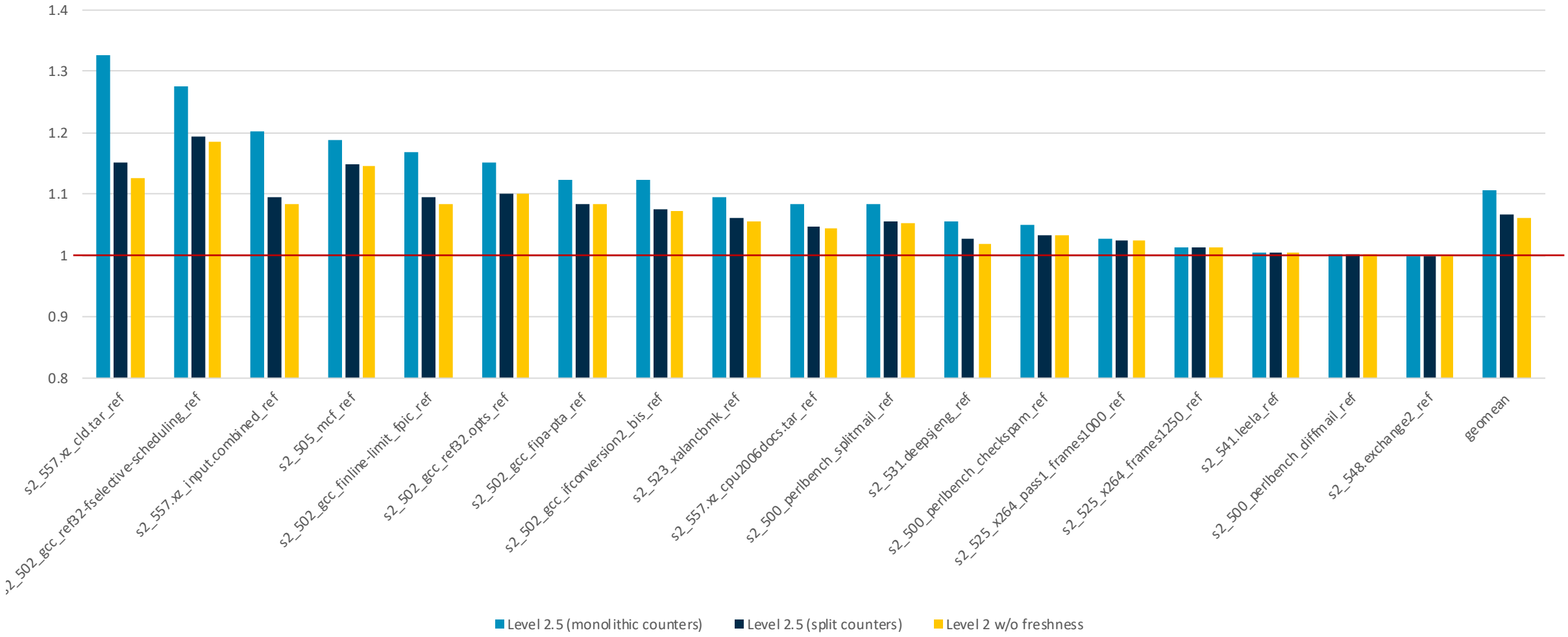
© 2019 Arm Limited

Levels 2A and 3A on Spec 2006



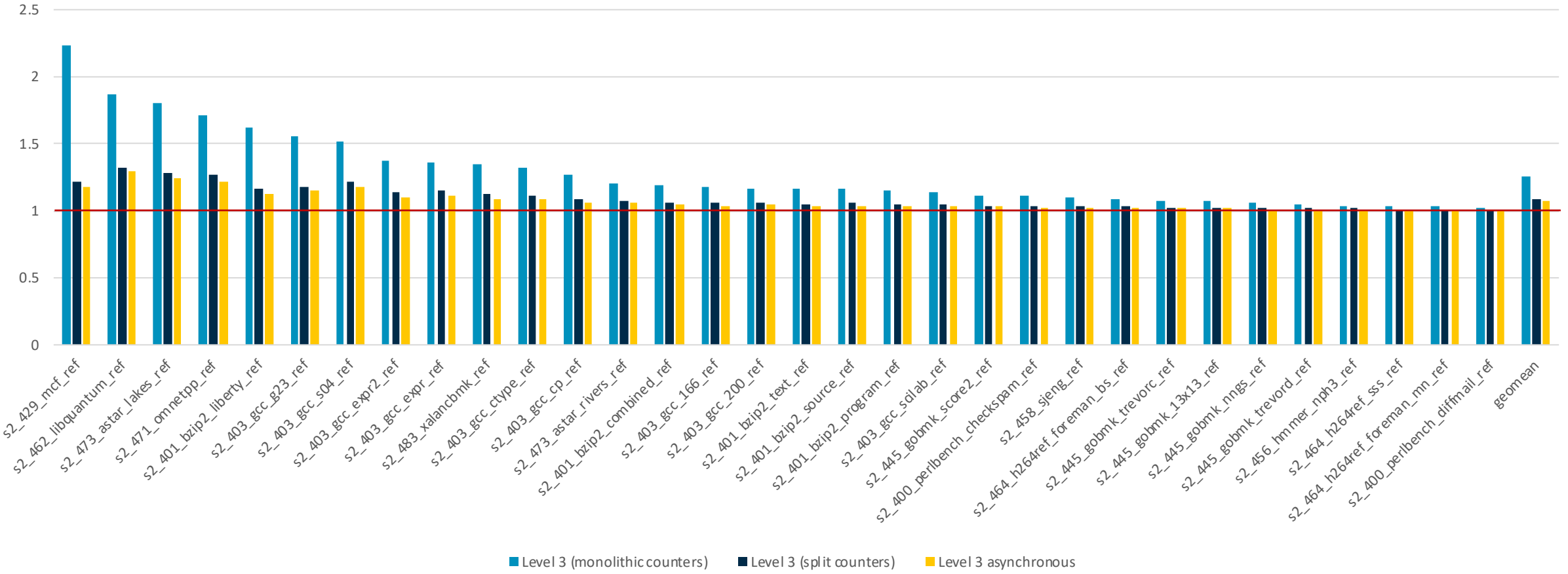
© 2019 Arm Limited
arm

Levels 2A and 3A on Spec 2017



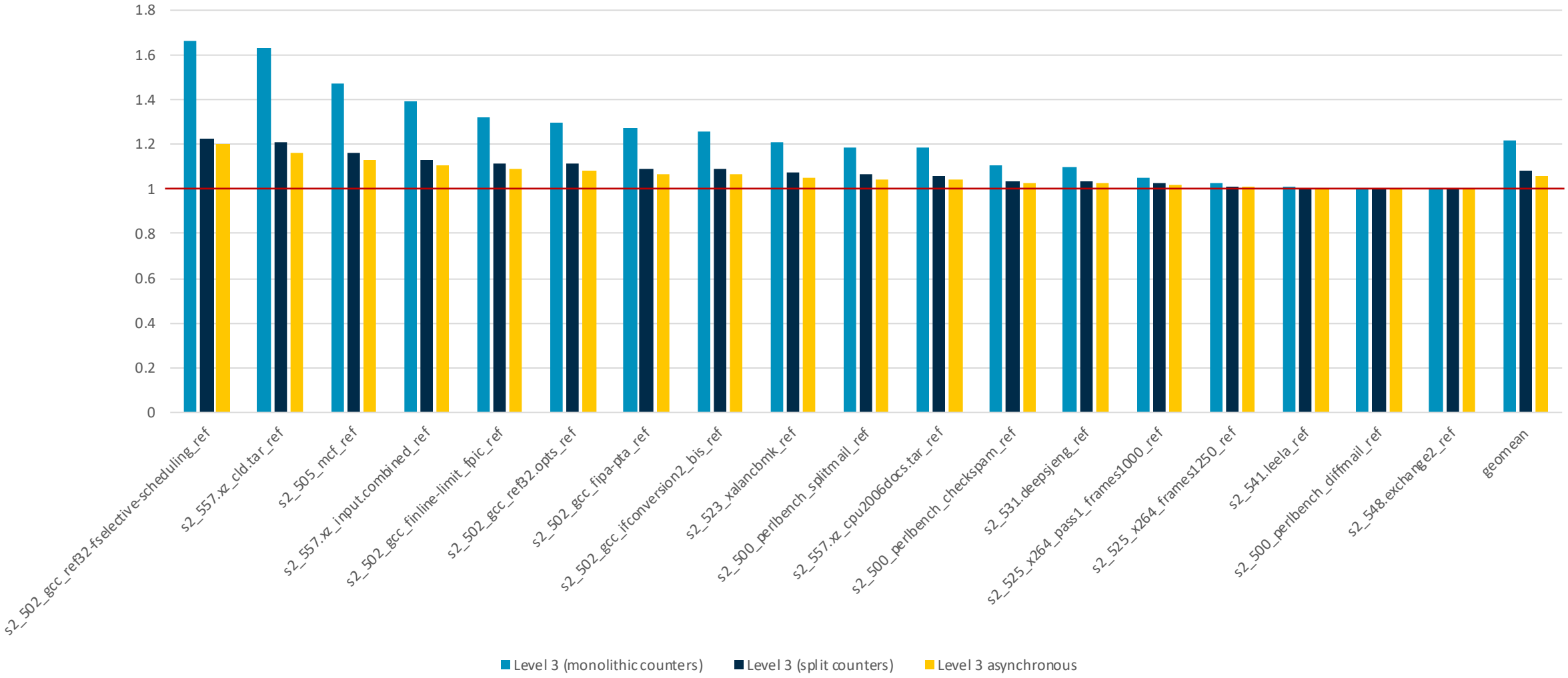
© 2019 Arm Limited

Level 3 on Spec 2006



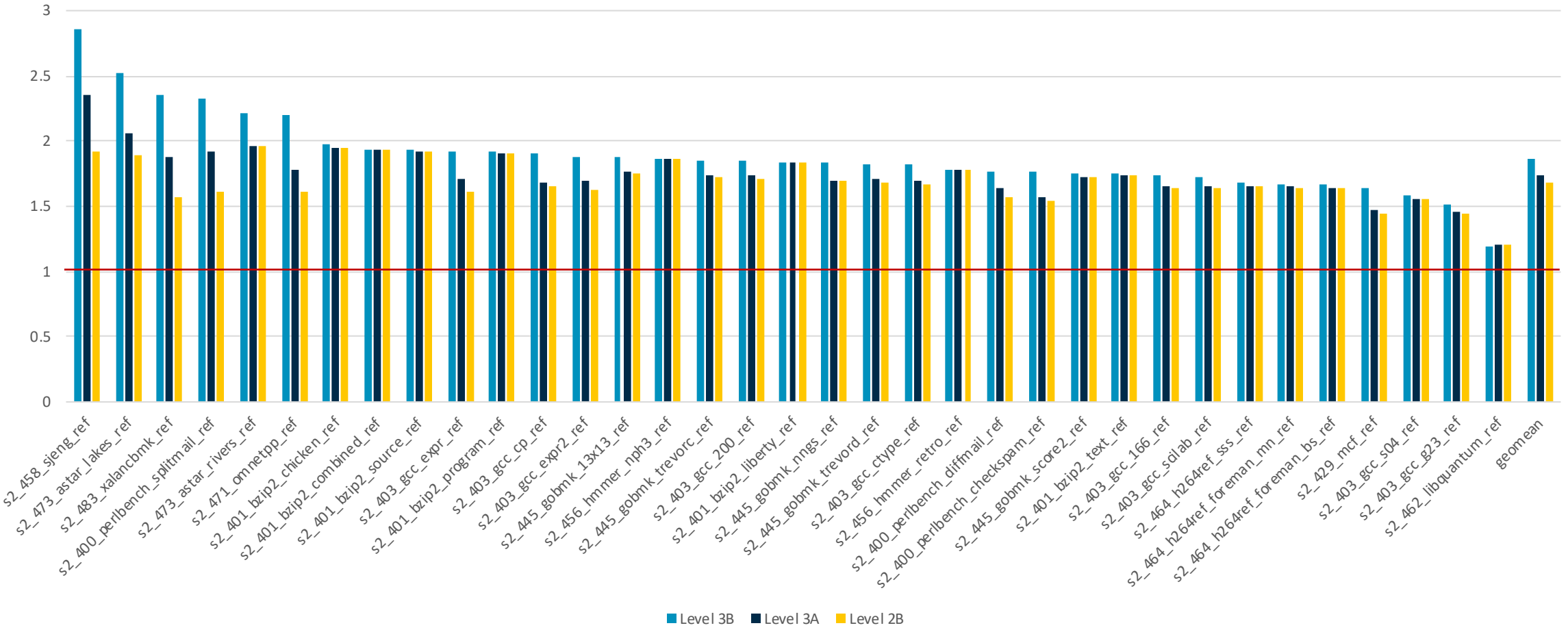
© 2019 Arm Limited
arm

Level 3 on Spec 2017



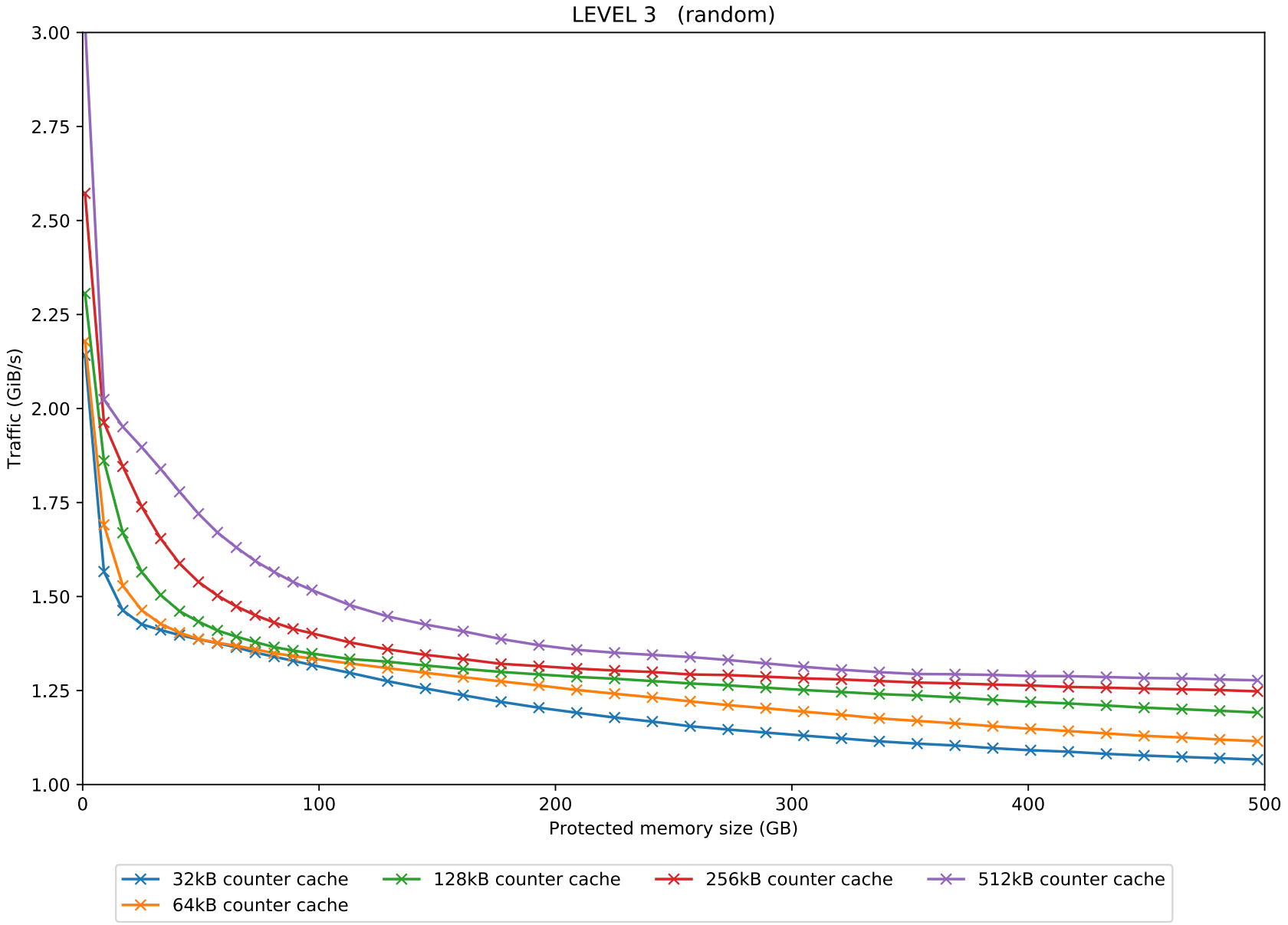
© 2019 Arm Limited

Memory Bandwidth Overhead



© 2019 Arm Limited

Bandwidth vs. Memory Size



© 2019 Arm Limited